
Serving WebObjects

To a large extent, WebObjects needs little attention once it is installed. However, there are certain basic administrative tasks that you'll need to know, such as how to install applications and start them up. In addition, you'll probably be concerned about your site's performance somewhere along the line, and you'll need to know how to determine an application's response times and what you can do to improve the response times.

This document contains information that a WebObjects system administrator needs to know. It begins by providing essential background information on WebObjects HTTP adaptors and how they are used to start WebObjects applications. Then it describes how to perform one-time setup of an application called Monitor, which you use to perform many of the necessary administrative tasks. After you read these first two sections, you can learn what the basic administrative tasks are and how to perform them.

Table of Contents

WebObjects HTTP Adaptors

- Configuration Files

- Adaptor Modes

- Installed HTTP Adaptors

Setting Up the Monitor Application

- Starting Up Monitor

- Initial Monitor Setup

Administrative Tasks

- Installing Applications

- Starting WebObjects Applications

- Monitoring Application Activity

- Performance Testing

- Improving Performance

- Periodically Shutting Down the Application

- Load Balancing

- Increasing the Listen Queue Depth

- Installing and Configuring NSAPI Adaptors

- Installing and Configuring the ISAPI Adaptor

- Starting Up Multiple Monitor Instances

Related Topics

Other WebObjects documents might be of interest to system administrators:

- *Installation Guide*: Includes system requirements, compatibility information, and location of the WebObjects Home Page. (The

Installation Guide is printed and included with the WebObjects CD-ROM or can be downloaded from NeXTanswers; it is not online).

- *Post-Installation Instructions*: Describes how to verify the installation and troubleshoot if WebObjects applications do not run.

WebObjects HTTP Adaptors

A key part of WebObjects administration is dealing with adaptors. This section provides a little background material on what a WebObjects HTTP adaptor is, how it works, and how you can configure it to suit your needs.

A WebObjects HTTP adaptor (called *WebObjects adaptor* or sometimes *HTTP adaptor*) routes client requests processed by an HTTP server to WebObjects applications and returns the response to the server, which sends it back to the client. WebObjects makes available several adaptors, of which only one can be active with a particular server at a time. Every transaction with a WebObjects application uses the currently active adaptor.

However, the relationships between adaptor and application are (potentially) many-to-many. Multiple instances of the same WebObjects application can run on the same machine or a variety of machines and communicate with the same adaptor. In addition, multiple HTTP servers can be running on the same machine or on different machines; each server can have its own adaptor, each with its own constellation of application instances. Although there can be only one active HTTP adaptor per HTTP server, an application can concurrently communicate with other types of adaptors, such as an adaptor that uses Distributed Objects or a secure-socket adaptor.

There are two general types of HTTP adaptors:

- The CGI adaptor, an executable file named `WebObjects` or `WebObjects.exe` which resides in the host HTTP server's "cgi-bin" directory. This adaptor is available on all supported platforms. It is generic in that it works with any HTTP server conforming to the Common Gateway Interface (CGI).
- API-based adaptors, that is, adaptors based on APIs specific to particular web server. Three NSAPI adaptors, all based on the Netscape Server API (2.0, 2.0.1, and 3.0), are available on all supported platforms except Mach. A WebObjects adaptor based on Microsoft's Internet Information Server API (ISAPI) is also supported. The API-based adaptors have a performance advantage over CGI adaptors in that the associated server can dynamically

load the adaptor; servers using CGI adaptors, on the other hand, spawn a new adaptor process for each request and kill the process after the response is provided.

When WebObjects is installed, the CGI adaptor is made active by default. To use an API-based adaptor, you must specifically activate it. Activating the API-based adaptor deactivates the CGI adaptor for a particular server. See “Installing and Configuring NSAPI Adaptors” and “Installing and Configuring the ISAPI Adaptor” for further details.

Configuration Files

WebObjects HTTP adaptors use configuration files to locate WebObjects application processes. There are two types of configuration files: public and private.

- The *public configuration file* is `NeXT_ROOT/NextLibrary/WOAdaptors/Configuration/WebObjects.conf`. (*NeXT_ROOT* is defined at system installation time.) This file tells the adaptor what applications are (or should be) running and allows the adaptor to balance transactions among different instances of the same application. You create the public configuration file using the Monitor application as described in the section “Initial Monitor Setup” in this guide.

In general, you want one public configuration file per site. That means if you have multiple machines running WebObjects, you should access all WebObjects applications through a single machine that is running the HTTP server and that contains the public configuration file.

If you have multiple HTTP servers running on a single machine, they all share the public configuration file. If you want each server to have its own configuration file, you can install one `WebObjects.conf` file in each server’s configuration directory.

- A *private configuration file* is also named `WebObjects.conf` and is located in the temporary directory of the system (`/tmp` for Mach, Solaris, and HP-UX platforms or the directory specified by the `TEMP` environment variable on the Windows NT platform). If the WebObjects adaptor cannot find the public configuration file or if it cannot find the requested WebObjects application in the public configuration file, it searches the private configuration file.

A new private configuration file is created automatically any time a WebObjects application is started and a private configuration file doesn’t exist. The adaptor contacts only one instance of an application

in the private configuration file; if you manually start HelloWorld and it's already been started, the entry for HelloWorld in the file is overwritten. (The old process will continue to run, but cannot be contacted.) The adaptor also cannot contact a remote instance of an application using the private configuration file.

The contents of the private configuration file are essentially the same as those of the public configuration file, except that the contents are stored as C structures and so cannot be directly modified. This file should only be modified by the WebObjects adaptor itself.

Adaptor Modes

All WebObjects adaptors route incoming requests to WebObjects applications in one of three modes:

1. Load-balancing between concurrent instances of the same application specified in the public configuration file
2. Choosing an application from the private configuration file
3. Autostarting an application

The active adaptor tries to contact the requested application by going through the modes in the preceding order.

Load Balancing: When the client request tries to contact an application, the active WebObjects adaptor first checks the public configuration file for an application matching the specification in the URL. Load balancing typically occurs only for the first request of a session if the application stores state in the server. Afterwards, the application resolves the URL so that page navigation will always occur in the context of the same application. But if the application stores state on the page or in cookies, true load balancing will be performed for *each* request.

Private Configuration File: If the adaptor cannot find a public configuration file, it attempts to resolve the URL against entries in the private configuration file. If the adaptor finds a matching entry but cannot contact it (for example, the application has been stopped), the adaptor deletes the “dangling” entry from the private configuration file and autostarts the application.

Autostarting: If there is no public configuration file and the adaptor fails to find an application matching the client's request in the private configuration file, it tries to start the application itself. The adaptor first searches in specific locations in the document root of the HTTP server and then in *NeXT_ROOT/NextLibrary/WOApps* for a WebObjects application (one with a *.woa* extension) that matches the specification in the request URL. If it finds one, it invokes the *WODefaultApp*

executable, or if the application is compiled, it invokes the executable in the application wrapper itself. This invocation starts the application on the HTTP server machine and registers it in the private configuration file, thereby allowing the adaptor to contact the new application instance. If the adaptor cannot find the requested application (for example, there's a typo in the URL), it returns a list of the applications it can find.

Note that if the public configuration file *NeXT_ROOT/NextLibrary/WOAdaptors/Configuration/WebObjects.conf* exists, no applications are ever autostarted. Also note that adding applications to the Monitor as described in the section "Initial Monitor Setup" creates the public configuration file. Thus, if you are using the Monitor application, autostarting is disabled.

Installed HTTP Adaptors

When WebObjects is installed, the following adaptors are put in *NeXT_ROOT/NextLibrary/WOAdaptors* along with source code. Note that only the CGI adaptor is installed on Mach, since neither Netscape's nor Microsoft's servers have been ported to this platform. Also, no ISAPI binary file is installed on Solaris or HP-UX platforms (only source code).

The following table summarizes the adaptors provided with WebObjects.

	Server	NextLibrary Location	Executable
CGI	many	WOAdaptors/CGI	WebObjects[.exe]
NSAPI2	Netscape 2.0 FastTrack (httpd) Enterprise (https)	WOAdaptors/NSAPI/2.0	WebObjects-NSAPI.dll or WebObjects-NSAPI.so
NSAPI2.0.1	Netscape 2.0.1 FastTrack (httpd) Enterprise (https)	WOAdaptors/NSAPI/2.0.1	WebObjects-NSAPI.dll or WebObjects-NSAPI.so
NSAPI3	Netscape 3.0 FastTrack (httpd) Enterprise (https)	WOAdaptors/NSAPI/2.0	WebObjects-NSAPI.dll or WebObjects-NSAPI.so
ISAPI	Microsoft Internet Information Server IIS 1.0 IIS 2.0 (NT Server 4.0) IIS 3.0 (NT Server 4.0) Peer Web (NT WS 4.0)	WOAdaptors/ISAPI	WebObjects-ISAPI.dll

Setting Up the Monitor Application

You can perform many essential administrative tasks using a WebObjects application named Monitor. Monitor helps you create a public configuration file, start and stop applications, perform load balancing, and set up applications for optimal performance. Before you can use Monitor, you need to perform some initial setup. After that, you can use it to easily administer your own WebObjects applications.

Note: Previous versions of WebObjects distributed Monitor as an example application. To use Monitor, you had to recompile your applications using a special version of WOApplication. This WOApplication class is incompatible with the Monitor application distributed in WebObjects 3.5. If you want to be able to Monitor these existing applications, you'll have to recompile them on WebObjects 3.5.

Starting Up Monitor

To start up Monitor on Windows NT, choose Monitor from the WebObjects program group in the Start menu.

To start up Monitor on any other system, do the following:

1. Open a command shell window.
2. Enter these commands:

```
> cd /NextLibrary/WOApps/Monitor.woa  
> Monitor
```

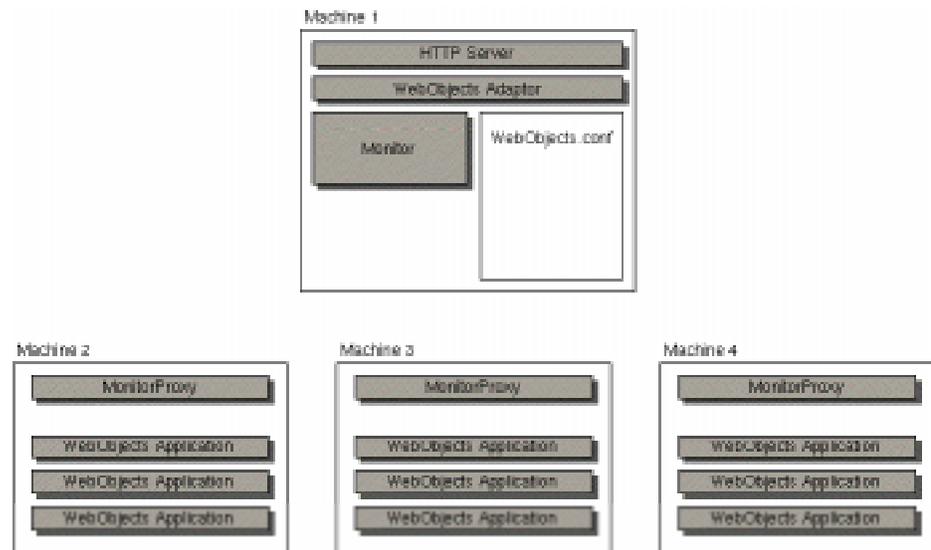
The Monitor application launches in your default web browser and displays this screen:



Which Copy of Monitor Should I Use?

If you have multiple machines running WebObjects, you can administer them all from a single instance of Monitor on a single machine. It's recommended that you use the copy of Monitor that's installed on the same machine as your HTTP server and WebObjects adaptor. This is because the main purpose of the Monitor is to maintain the public configuration file (`WebObjects.conf`), which the WebObjects adaptor uses to find running instances of WebObjects applications. The best way to achieve sharing of the `WebObjects.conf` file is to have your HTTP server and Monitor run on the same machine. The alternative—having Monitor on a separate machine—would require that the two machines share a file system through network access.

Monitor can communicate with WebObjects applications running on remote hosts; however to launch applications on remote hosts it uses a lightweight daemon named `MonitorProxy`. For example, the following figure shows a WebObjects site spread across four machines. One machine contains the HTTP server, and the other three machines contain WebObjects applications. You would run the Monitor application on the machine containing the HTTP server. That Monitor application would use the `MonitorProxy` daemons on the other three machines to launch applications. All other communication goes directly between Monitor and the WebObjects application.



Starting Up MonitorProxy

As mentioned in the previous section, if you have a multiple machine configuration, you need to run Monitor on one machine and run **MonitorProxy** daemons on the other machines. You can start up **MonitorProxy** daemons the same way you start up Monitor:

To start up **MonitorProxy** on Windows NT, do the following:

1. Navigate to the directory `NeXT_ROOT/NextLibrary/WOApps/Monitor.woa` in the Explorer.
2. Double-click **MonitorProxy.exe**.

To start up **MonitorProxy** on any other system, do the following:

1. Open a command shell window.
2. Enter these commands:

```
> cd $NEXT_ROOT/NextLibrary/WOApps/Monitor.woa
> MonitorProxy
```

You probably want to set up your system so that **MonitorProxy** starts up at system boot time.

Note: If you're running Monitor on a Solaris or HP-UX machine, you must start up **MonitorProxy** on that machine as well. As you'll learn later, Monitor requires the local **MonitorProxy** to start up applications on Solaris and HP-UX.

Initial Monitor Setup

In normal operation, when you start up Monitor, all of the WebObjects applications on your site are displayed under the Declared Apps heading, and the right frame shows information about the application you select. (Or if you set it up properly, you'll see a login panel and after you log in you'll see all of your WebObjects applications.) However, the first time you've started the Monitor, no applications are displayed. You need to configure the Monitor for use on your site.



Click here to configure the Monitor application.

To configure the Monitor application, do the following:

1. Click the Options button.

The right frame shows configuration information.

2. Look at the first section of the Configuration options. They should look like this:

Local host name:	<input type="text" value="stingray"/>
Admin user name:	<input type="text"/>
Admin password:	<input type="password"/>
Admin password again:	<input type="password"/>

Make sure each option has the appropriate setting as described below:

Local host name

The name of the server Monitor is running on.

Admin user name

Protects the Monitor from being used by all users except the one specified here. If you enter a user name in this field and that user's password in the "Admin password" and "Admin password again" fields, Monitor runs in a protected mode. When you restart Monitor, it displays a login page instead of the typical first page.

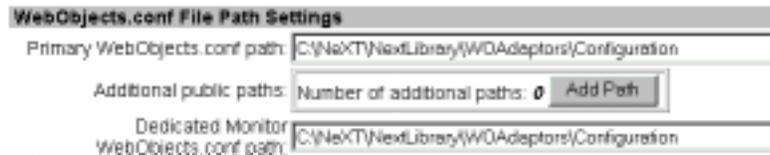
Admin password

If you entered an administrator name, enter the administrator account's password here.

Admin password again

Repeat the administrator password here.

The second set of options affects the configuration file. Make sure each option has the appropriate setting as described below:



Primary WebObjects.conf path

The location of the public configuration file. Most of the time, this should be set to *NeXT_ROOT/NextLibrary/WOAdaptors/Configuration*. (You only need to change this if you move the configuration file.)

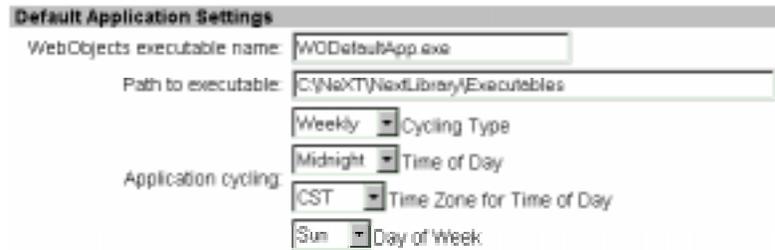
Additional public paths

If you have more than one server and each server has its own public configuration file, click the Add Path button and specify the location of the other configuration file(s). Servers can share the configuration file in *NeXT_ROOT/NextLibrary/WOAdaptors/Configuration*, so usually this setting is not necessary.

Dedicated Monitor WebObjects.conf path

The location of a configuration file used only by Monitor. This file is used only if you want to start multiple instances of Monitor, which is rare. See "Starting Up Multiple Monitor Instances."

The third set of options displays default setting for each application that you'll add to the Monitor. Make sure each option has the appropriate setting as described below:



Default Application Settings

WebObjects executable name:

Path to executable:

Application cycling: Cycling Type

Time of Day

Time Zone for Time of Day

Day of Week

WebObjects executable name

The executable that you want to use as the default executable for all applications you add to Monitor. By default, this is set to `WODefaultApp` or `WODefaultApp.exe`.

Path to executable

The path to the executable name you specified in the field above.

Application cycling

Default settings for the periodic shutdown of applications. For more information, see “Periodically Shutting Down the Application” in this guide.

The fourth set of options you rarely have to change. They are used to build the WebObjects application URLs that you use when you click an application’s hyperlink to access one of its running instances.



Application Startup Defaults

HTTP server host:

Adaptor key:

Protocol name:

HTTP server host

The host name of your HTTP server.

Adaptor key

The name of your WebObjects adaptor. Usually, this is `WebObjects` or `WebObjects.exe`. If you use the NSAPI adaptor or ISAPI adaptor, you may have to change the name here.

Protocol name

Should normally be set to `http`.

3. Click the Save Changes button at the bottom of the page to record your changes.
4. In the Main Menu area, click the Add button.



Click here to add applications.

The right frame displays the Add Application Panel, which looks like this:

Enter the application's name here.

Enter its path here.

Enter the host name here.

5. For each application that you want to administer using Monitor, do the following:
 - a. Type the application's name in the Application Name field. The application name is the path to the application relative to `<DocRoot>/WebObjects`.
 - b. Type the path to the application executable in the Application Path field (for example `C:\Next\NextLibrary\WOApps\MyApp.woa`)
 - c. Type the host name of the machine where the application is located in the Host name field.

d. Click the Add to Declared Apps button.

Monitor updates the port number and instance number fields for you.

When you use the Add Application panel, you are creating the public configuration file *NeXT_ROOT/NextLibrary/WOAdaptors/Configuration/WebObjects.conf*. This has a few implications that might not be immediately apparent:

- Autostarting is disabled.

Recall that when the public configuration file exists, autostarting is disabled. You can still start applications manually from the command line, but applications won't autostart when you type a URL in the browser.

- You won't get an application listing if you enter `http://localhost/cgi-bin/WebObjects/` or if you make a typo in an application URL.

This feature depends on autostarting, so it is disabled when autostarting is disabled.

- If you start an application from the command line, you must use the `-n` option to specify an instance number.

Monitor associates an instance number with all applications that you declare in it, instead of just applications that perform load balancing among multiple instances. If you manually start an application and you want it to connect to the Monitor, you must specify the instance number that Monitor set up for you. For more information, see "Starting Up Applications From the Command Line" in this guide.

Administrative Tasks

This section covers typical administrative tasks that you may need to perform:

- Installing Applications
- Starting WebObjects Applications
- Monitoring Application Activity
- Performance Testing
- Improving Performance
- Periodically Shutting Down the Application

Load Balancing
Increasing the Listen Queue Depth
Installing and Configuring NSAPI Adaptors
Installing and Configuring the ISAPI Adaptor
Starting Up Multiple Monitor Instances

Installing Applications

You can use the developer application Project Builder to deploy WebObjects applications. When an application is ready to be deployed, do the following in Project Builder:

1. Click the inspector button to open the Build Attributes Inspector. In the Install in field, type `$(NEXT_ROOT)/NextLibrary/WOApps`.

If you're installing a framework, type `$(NEXT_ROOT)/NextLibrary/Frameworks`

2. If your project contains web server resources, go to the `Makefile.preamble` file under Supporting Files. Uncomment the following macro:

```
INSTALLDIR_WEBSERVER
```

3. In the Project Build panel, click the checkmark button to bring up the Build Options panel.
4. Choose `install` as the build target, and close the Build Options panel.
5. Click the Build button to start the build and installation process.

Assuming that your application is named `MyApp.woa`, this procedure installs these directories:

```
NeXT_ROOT/NextLibrary/WOApps/MyApp.woa  
  MyApp[.exe]  
  Resources/  
  WebServerResources/  
<DocRoot>/WebObjects/MyApp.woa  
  WebServerResources/
```

As discussed previously in the section “Adaptor Modes,” when the client tries to contact an application, the adaptor first looks for a public configuration file that names the application, then for a private configuration file that names the application, and then for an executable in `<DocRoot>/WebObjects` and `NeXT_ROOT/NextLibrary/WOApps`. Thus, you can install the entire directory under `<DocRoot>/WebObjects`, but doing so presents a security problem if you have scripted components. Any client can access any file under the document root,

which means that if you install scripted components under the document root, you are exposing source code to outside users.

Instead, it is recommended that you install most of the application in *NeXT_ROOT/NextLibrary/WOApps* and install only the web server resources under the document root. It is also recommended that you install the application directly in the *<DocRoot>/WebObjects* directory rather than in a subdirectory. If you install in a subdirectory, your application can still run but cannot find image files unless you provide the application path on the command line. For more information, see “Starting Up Applications From the Command Line” in this guide.

Starting WebObjects Applications

As described previously, there are two ways to start an application: autostart and manual. Autostarting applications occurs when the user types the application’s URL in a web browser; the WebObjects adaptor looks for a running instance of that application and starts one if it cannot find one. Autostarting is not recommended for deployment, It’s convenient but when you allow autostarting, it becomes more difficult to monitor the applications progress. Instead, you can use the Monitor application’s interface to start applications. (The application will sit idle until a user tries to access it.) Another alternative is to start applications from the command line. You might start from the command line when you need to see the debugging messages written to standard output. This section describes how to manually start an application using the Monitor or the command line.

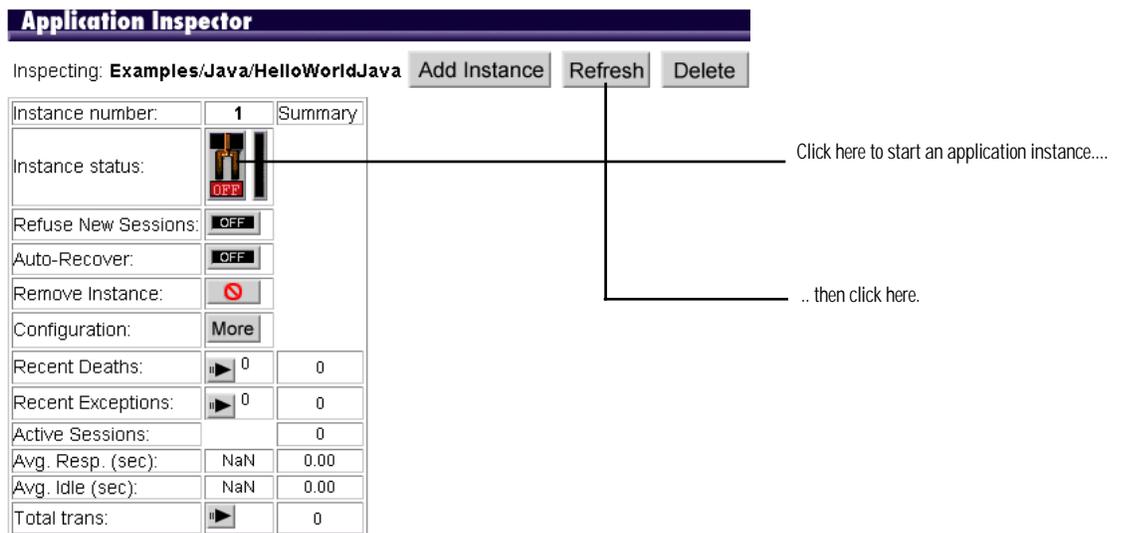
Starting Applications Using Monitor

To start up a WebObjects application in the Monitor, do the following:

1. Locate the application in the list in the left frame of the browser window.
2. Click the inspector button to the left of the application name.



3. Click the On/Off button located in the Instance Status field.



Monitor starts an instance of your application by creating a new task with the executable and any arguments you have set Monitor to use for that instance. Monitor starts your application in one of two ways, depending on whether the application is on a different host.

- If the application is on a different host, it tries to locate a running **MonitorProxy** on that host. If it finds a **MonitorProxy**, it passes the application arguments to it. If it can't find a **MonitorProxy**, it will fail to start the application.

- If the application is to run on the same host as the Monitor, Monitor looks for a `MonitorProxy` on its own machine. If a `MonitorProxy` is found, it passes any arguments to the `MonitorProxy` and uses it to start the application. If no `MonitorProxy` is found, Monitor starts your application itself.

Note: On Solaris and HP-UX platforms, Monitor cannot start applications itself. It must use the `MonitorProxy`, even if the application is on the local machine.

To learn whether the application startup was successful, you must click the Refresh button at the top of the window. If the startup was successful, the On/Off button is set to On, and the application's name in the left frame has become a hyperlink. Why do you have to click the Refresh button? As soon as a WebObjects application starts up, it notifies the Monitor immediately, but the Monitor cannot refresh the web browser itself without a request from the web browser.

If the button is not set to On, either the application did not get a chance to start before you refreshed the page display, or the application did not start because of an error. If the application has a very involved initialization, you may have to wait several seconds before clicking Refresh shows the application as running.

This procedure starts an instance of the application but does not display it in the web browser. You can access an application instance by clicking the hyperlink in the left frame, but you are not guaranteed to get the instance that just started; if multiple instances are running, the adaptor performs load balancing to process your request.

Starting Up Applications From the Command Line

The syntax for starting a WebObjects application from a command shell window is:

```
AppExecutable [-debug ON|OFF] [-browser ON|OFF] [-m ON|OFF] [-mhost hostname|subnet] [-c] [-d DocumentRoot] [[-a AdaptorClass] [-n InstanceNumber] [-p PortNumber] [-q ListenQueueDepth]] [ApplicationName]
```

AppExecutable

The name of the WebObjects application executable to run. You should enter the command from the directory containing the executable. Compiled applications should either be located in `NeXT_ROOT/NextLibrary/WOApps` (recommended) or under `<DocRoot>/WebObjects`. For scripted applications, go to the

application's `.woa` directory and execute `WODefaultApp`, which is located in `NeXT_ROOT/NextLibrary/Executables`.

`-debug ON|OFF`

Sets whether the application prints messages to standard output during startup. By default, this option is ON. If this option is ON, the application prints messages such as the following:

```
Aug 22 18:08:46 WODefaultApp[423] Application Path:
/NextDeveloper/Examples/WebObjects/HelloWorld
Aug 22 18:08:50 WODefaultApp[423] Application Base URL:
/WebObjects/Examples/HelloWorld
Aug 22 18:08:50 WODefaultApp[423] Application Name:
Examples/HelloWorld
Aug 22 18:08:51 WODefaultApp[423] Reading WebServer configuration
from /NextLibrary/WOAdaptors/Configuration/WebServerConfig.plist.
Aug 22 18:08:51 WODefaultApp[423] Opening application's URL in
Browser:
http://localhost/cgi-bin/WebObjects/Examples/HelloWorld
Aug 22 18:08:54 WODefaultApp[423] waiting for requests...
```

`-browser ON|OFF`

Sets whether the application automatically starts up the web browser. The default is ON. If this option is ON, the browser automatically opens a new browser window (starting up the browser if necessary) with the WebObjects application's URL.

`-m ON|OFF`

Enables or disables monitoring. The default is ON. If this option is ON and you manually start up an application, the application tries to find a running Monitor. If it finds one, Monitor can automatically locate the application and display information about it, provided an instance number is given with the adaptor's `-n` option as described below.

`-mhost hostname | subnet`

The application tries to find a running Monitor on the machine named *hostname* instead of on the local machine. If `subnet` is used, the application looks for a running Monitor in its network subnet.

`-c`

Requests that the application cache component definitions (templates) instead of reparsing HTML and declaration files upon each new HTTP request. By default, applications do not cache component definitions. This setting ensures that during development of scripted applications programmers can modify a component's logic and see the result without having to relaunch the application. If you are deploying applications, however, you should turn on component-definition caching by specifying this flag when you launch the application.

-d *DocumentRoot*

The document root of the server, which can be different from the document root specified for a given web server. If you use this option, you must also specify the *ApplicationName* option. If you don't specify a document root, it is taken from the configuration file *NeXT_ROOT/NextLibrary/WOAdaptors/Configuration/WebServerConfig.plist*.

-a *AdaptorClass*

The class of an adaptor that the application will use to communicate with the server. You can specify multiple adaptors, as long as they are of different types. (For example, you could have a separate adaptor with its own port for communicating directly with Java applets on the browser.) If you specify multiple HTTP adaptors, only the last one specified will be made the active one.

The subsequent three arguments belong to the adaptor specified in *AdaptorClass*; the first two of these are used in load balancing: You cannot specify adaptor arguments unless you specify an adaptor class.

-n *InstanceNumber*

A positive integer that uniquely identifies an application instance with which the adaptor will communicate. If you do not specify an instance number, the adaptor specified in *AdaptorClass* creates one using random number generation. If a URL does not specify the instance number, the application is presumed to run on the server machine as a single instance application, as if it had been autostarted. If you specify *AdaptorClass* for the purpose of load balancing, you must specify an instance number.

Note that if you intend to use Monitor to administer the application, you must use the -n option. Monitor always assigns instances to the applications it knows about. If you start up an application without the -n option, its instance number is nil, and Monitor is not able to connect to it.

-p *PortNumber*

Specifies the socket port used to communicate with an application instance. Port numbers must be over 1024 since numbers between 0 and 1024 are reserved. If you specify *AdaptorClass* for the purpose of load balancing, you must specify a port number.

-q *ListenQueueDepth*

Specifies the queue depth on a TCP/IP socket at the entrance of the application. The default listen queue depth is 4, meaning that while the application process is handling a request, up to four other requests can be in the socket buffer before the socket starts refusing them. If the application is expected to experience "spikes" in its processing load, it might be a good idea to increase the listen queue depth. Increasing this default does not nec-

essarily improve performance or allow the application to serve more requests at sustained high loads. For more information, see “Increasing the Listen Queue Depth” in this guide.

ApplicationName

Specifies the application name, which is the directory path relative to `<DocRoot>/WebObjects`. This argument is required when you use the `-d` option to specify the document root.

Examples

The following example starts the scripted application TimeOff on Windows NT:

```
> cd <DocRoot>\Examples\WebScript\TimeOff.woa
> WODefaultApp.exe Examples/WebScript/TimeOff
```

The following example starts a compiled WebObjects example application on Mach, assigning it the default HTTP (CGI) adaptor and specifying port and instance numbers for that adaptor. Because an instance number is specified, if a Monitor is running on that machine, it can display application activity and shut down the application.

```
> HelloWorldCompiled -d /NextLibrary/WebServer/htdocs -a WODefaultAdaptor
-n 1 -p 3000 -q 50 Examples/ObjC/HelloWorldCompiled
```

Notes

The web server uses the `<DocRoot>` and `ApplicationName` arguments to build URLs, so you should use forward slashes as opposed to a backslashes when specifying these arguments.

As a convenience, you might create a shell script that starts WebObjects applications when the server machine is booted. You also might create another shell script that you can run at the command line to start applications.

Monitoring Application Activity

There are several ways you can obtain information about the applications running on your server. You can use the Monitor application, analyze logs kept by the application and the adaptor, and check the application’s statistics page.

Using the Monitor to Obtain Application Information

The Monitor application can provide you with information about all of your running WebObjects applications.

Application Inspector

Inspecting: **Examples/WebScript/HelloWorld** Add Instance Refresh Delete

Instance number:	1	Summary
Instance status:		
Refuse New Sessions:	<input type="checkbox"/> OFF	
Auto-Recover:	<input type="checkbox"/> OFF	
Remove Instance:		
Configuration:	More	
Recent Deaths:	 0	0
Recent Exceptions:	 0	0
Active Sessions:	0	0
Avg. Resp. (sec):	0.00	0.00
Avg. Idle (sec):	0.00	0.00
Total trans:	 0	0

- The left side of the screen shows you how many WebObjects applications you have on the system and which ones currently have at least one instance running. If an instance is running, the application name is a hyperlink.
- Click the inspector button, and the Application Inspector appears in the right frame. It shows you how many instances you've configured for that application (that is, if you're set up to do load balancing), which instances are running, whether the instance has encountered an exception, and if the instance has recently died.
- Click the More button to see detailed information about an application instance in the bottom part of the right frame. This page shows you the arguments used to start the application: the executable name, host name, document root, port number, and instance number. It also shows you if periodic application shutdown is enabled or disabled and, if enabled, under what circumstances the application will shut down. (For more information, see "Periodically Shutting Down the Application" in this guide.)
- In the Application Inspector, you can click the arrow button in the Transactions field to obtain information about the number of transactions received, the response times for each transaction, and the average response time.
- In the Application Inspector, if the number in the Recent Deaths field is greater than 0, you can click the arrow button in the field to obtain

more information about when the application died and how many times the application died.

- In the Application Inspector, if the number in the Recent Exceptions field is greater than 0, you can click the arrow button in that field to obtain information about the exception. This screen shows you which exception occurred and attempts to describe which part of the application was executing at the time of the exception.

Logging and Analyzing Application Activity

WebObjects applications can record information in a log file that can be analyzed by a Common Log File Format (CLFF) standard analysis tool. Applications do not maintain this log file by default; log file recording must be enabled programmatically. If enabled, the application records a list of components accessed during each session. By default, only component names are recorded, but programmers may add more information.

Run any CLFF standard analysis tool to analyze the information in the log.

Logging and Analyzing Adaptor Activity

If an adaptor sees that a file named `logWebObjects` exists in the temporary directory, it will log its activity in `WebObjects.log` in that same directory. Logging adaptor activity significantly decreases performance. Use this feature only if you suspect something is wrong.

The temporary directory depends on the platform:

- `/tmp` on Mach, Solaris, and HP/UX
- The directory indicated by the `TEMP` environment variable on Windows NT

You can analyze the information in the log to find out such things as which applications are being requested, which applications are being autostarted, and what the HTTP headers of requests are. You can also use the log to verify if adaptors are properly configured for load balancing. For example, the following excerpt includes a warning message printed when the adaptor cannot find the `WebObjects.conf` file in the expected location.

```
INFO: -- WOServerAdaptor: Load Balancing for Examples/TimeOff
WARN: -- WOServerAdaptor: "No such file or directory" occurred while
opening the configuration file C:\NETSCAPE\ns-home\httpd-
80\config\WebObjects.conf
```

The procedure is:

1. Start a command shell window (on NT use the Bourne Shell in the WebObjects program group).
2. Change to the temporary directory (using the `cd` command).
3. Enter the following command to create the `logWebObjects` file:

```
touch logWebObjects
```

4. Enter the `tail` command to print the current activity in the adaptor to standard output (the shell window):

```
tail -f WebObjects.log
```

Accessing the Application Statistics Page

WebObjects applications record statistics about themselves while they run. You can access the WOStats page while the application is running to obtain access to these statistics. Most WebObjects applications have a WOStats page automatically included in the application. Access the WOStats page with a URL like the following:

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

If there are multiple instances, specify the instance number as well:

```
http://myhost/cgi-bin/WebObjects/MyWebApp.woa/-/WOStats.wo/-/1/myhost
```

The “1” just before the last occurrence of “myhost” is the instance number.

Statistics For Examples/CyberWind On Host gluon

Application Statistics

Transactions	7.00
Avg. Transaction Time	1.82
Avg. Idle Time	20.26
Moving Avg. Transaction Time	1.82
Moving Avg. Idle Time	20.26
Sample Size for Moving Avg.	100
Started at	06:02:09 (-0700 PDT) on Mon, Sep 15 1997
Running time	0 days, 0 hours, 2 minutes, 37 seconds

Sessions Statistics

Avg. Session Life	0.53
Total Sessions Created	6.00
Peak Active Sessions	5.00
Avg. Transactions Per Session	1.00
Current Active Sessions	5.00

See the description of WOSTats in the *WOExtensions Reference* for more information about what the page displays.

Performance Testing

The WebObjects package comes with a special adaptor that allows you to record a session and a tool that helps you play back a recorded session. Using these tools, you can test your application setup to determine if you have the appropriate number of instances running, the appropriate amount of memory allocated, and so on.

Note: You cannot use the recording and playback tools on applications that use HTML frames.

To use the recording and playback performance testing tools, do the following:

1. Copy the adaptor `WebObjects-Recording` from `NeXT_ROOT/NextLibrary/WOAdaptors/CGI` to your web server's cgi-bin directory.
2. Use the Monitor to create an instance of the application.

3. Start the application instance you just created. To do so, open a command-shell window and enter the command line as shown in the section “Starting Up Applications From the Command Line.” Use the `-n` option (which must be used in conjunction with the `-a` option) to specify the instance number. Also use the `-browser OFF` option so that the application does not automatically launch in your web browser. For example, your command line might look like this:

```
MyApp -a WODefaultAdaptor -n instanceNumber -c -browser OFF
```

4. In your web browser, enter this URL:

```
http://localhost/cgi-bin/WebObjects-Recording/MyApp?file=completePath
```

where *completePath* is the directory in which you want to store the recorded session. The adaptor appends a `.rec` extension to the path you specify.

5. Using the web browser, run a session of your WebObjects application. You may want to record what you believe to be a typical session, or you may want to perform a session that would put a maximum load on your system. For example, you may want to record a session that performs as many database fetches as possible.

Important: During recording, only one user may be accessing the application. Your session must not include any backtracking to a previously displayed page. If you backtrack, you'll get unpredictable results.

As you run the application, the WebObjects recording adaptor records each request and response to a separate file in the directory you specified.

6. When you have finished the session, close the browser window. To prevent accidental calls to the WebObjects-Recording adaptor, remove it from your server's `cgi-bin` directory.
7. Open a command shell window and enter this command:

```
WOPlayback -R completePath.rec -H hostname
```

where *completePath.rec* is the directory that contains the recorded session and *hostname* is the name of the host on which you want to run the recorded session.

The WOPlayback tool plays the recorded session repeatedly until you explicitly stop it (for example, by pressing Control-C in a command

shell window). It is possible to run several versions of WOPlayback at the same time to put more load on the server.

If you want, you can specify other options to the WOPlayback tool as well. The following is a list of the available options:

-P *http_port*

The port number of your HTTP server (the default is 80).

-C *count*

Plays back the session *count* number of times instead of indefinitely.

-S *sleep_time*

The number of seconds to wait in between requests. The default is 0.

-p *adaptor_path*

Sends request using the *adaptor_path* instead of the recorded URL. For example, suppose you recorded a session using a Netscape server whose cgi-bin directory is named `cgi-bin` and you want to play it back using the Microsoft Internet Information Server, whose cgi-bin directory is named `Scripts` and has `WebObjects.dll` as the adaptor name. Your *adaptor_path* is

```
/Scripts/WebObjects.dll
```

If you'd like to improve the average response time that resulted from this test, read the next section, "Improving Performance" for guidelines on how to do so.

Improving Performance

Performance is a major concern of website administrators. This section provides a list of areas to check to achieve the maximum possible performance.

- Configure your operating system so that it delivers the best performance possible for your needs. Check your operating system's documentation and your web server's documentation for performance tuning information.
- When possible, use an API-based adaptor in place of the default CGI adaptor.

The API-based adaptors have a performance advantage over CGI adaptors in that the associated server can dynamically load the adaptor; servers using CGI adaptors, on the other hand, spawn a new adaptor process for each request and kill the process after the response is provided.

- Make sure that the applications are written to perform optimally.

The *WebObjects Developer's Guide* offers some suggested coding practices to improve performance.

- Enable component-definition caching for all applications.

Component-definition caching is off by default as a convenience for programmers debugging applications. When the application is deployed, component-definition caching should be enabled so that each component's HTML and declarations files are parsed only once per session. Component-definition caching can be enabled programmatically by sending `setCachingEnabled:` to the `WOApplication` object (in Java, `WebApplication`). You can also use the Monitor to enable caching by doing the following:

- a. In the Declared Apps list, click the inspector button for the application to display the application inspector.
- b. Click the More button to display the application instance inspector.
- c. Click the Component caching check box.
- d. Click the Save Settings button at the bottom of the frame.

- Shut down and restart application instances periodically.

Because no program is ever perfect, WebObjects applications may leak a certain amount of memory per transaction. For this reason, you should periodically shut down and start up each application instance as described in "Periodically Shutting Down the Application" in this guide.

- Perform load balancing or increase the listen queue depth to improve response time for a specific application.
 - If the response time is consistently slow, add more application instances so that the load is balanced among those instances. For more information, see the section "Load Balancing" in this guide.
 - If the response time is sometimes acceptable and sometimes slow, consider increasing the size of the listen queue, which holds requests awaiting processing. For more information, see the section "Increasing the Listen Queue Depth" in this guide.
- Consider changing the physical configuration of your system.

Determine the size of a single application instance (you can look this up on the application's `WOStats` page) and multiply that number by

the number of instances you intend to run on a given machine. The result is the amount of physical memory that should be installed on that machine.

If you can't add that much physical memory, increase the amount of virtual memory to cover the difference between the physical memory needed and the physical memory you have.

You can also try to reduce the size of the application instance by limiting the amount of state that it stores. Set the session time-out value to ensure that sessions expire after a reasonable length of time. Shut down and restart the application more often to reduce its size.

If you use WebObjects mainly for applications that access a database, you'll achieve the best performance with a dedicated database server and a separate server for WebObjects applications.

Periodically Shutting Down the Application

The longer an application runs, the more memory it consumes. As more memory is consumed, the server machine's performance begins to degrade. For this reason, you may find that performance is greatly improved if you occasionally stop an application instance and start it again.

The Monitor application lets you specify when each application instance should shut down. By default, WebObjects applications never shut down unless you manually shut them down (which you can do by clicking the On/Off switch in the Application Inspector Instance Status field).

To set up periodic application shutdown, do the following:

1. In the Application Inspector, click the More button to show the Application Instance Inspector in the bottom frame.
2. Scroll down until you see the check box labeled Scheduling Enabled. Click this check box.
3. Choose how the application should determine when to shut down and restart. There are several options:
 - **Application cycling.** You can have an application terminate itself at the end of a given period of time. To set this up, choose Daily, Weekly, or one of the numerical Hourly settings in the Application Cycling pop-up list, then, if applicable, set the time of day at which you want the application to shut down. (You should choose a time when a minimum number of users is trying to access the application. For example, if all of your users are on the West Coast of North America, you can set the application to shutdown at 2:00AM PST.)

If you want all applications to use the same settings, you can change the default shut-down time on Monitor's Configuration Options page. Press the Options button in the Main Menu and scroll until you see the same application shut-down options as described here. When you change the settings on this page, it affects any applications you add in the future; existing application settings are not affected.

- **Shut down and start at set times.** You can have an application terminate itself and start itself at a given date and time. Check the Future Shutdown check box and enter the time at which the application should shut down. If you want it to restart at a set time as well, click the Future Startup checkbox and enter the time at which you want it to start.

Tip: To have an application instance start up when you start up the Monitor, set it to start at a date in the past. This ensures that a dead instance of that application can be recovered. For example, you might have set an application instance to auto-recover from a shutdown. Suppose that the application crashes at a time when Monitor isn't running (because it has crashed as well). Your instance won't restart because Monitor isn't around to make it restart. If you schedule the application to start in the past, it will start as soon as a Monitor instance starts.

- **Session count.** An application can terminate if the number of active sessions falls below a certain number. Set this number in the Minimum Active Sessions field. Then click the check box labeled Enabled Inactivity Self-Kill.

If Minimum Active Sessions and Enable Inactivity Self-Kill are set, you can, if necessary, click Refuse New Sessions in the Application Inspector at any time while the application is running. When Refuse New Sessions is enabled, the application will not accept any requests from users it does not already know about. After all of the current sessions have expired, the application instance shuts down.

Using the session count to terminate an application is a much more graceful way of shutting down application instances. Scheduled shutdowns will terminate the application no matter how many users are accessing it at that moment. However, if you set Minimum Active Sessions, and Enable Inactivity Self-Kill, it allows all currently active sessions to complete before the application shuts down.

4. If you want the application to restart automatically after it is shut down, click the Auto-Recover button in the Application Inspector. If this setting is not enabled, you'll have to manually restart the application instance.
5. Click Save Settings.

You can programmatically set up an application to shut down in addition to setting it using the Monitor. If the two settings conflict, the Monitor settings override the application's settings. For more information, see the *WebObjects Developer's Guide*.

Load Balancing

You can improve the performance of a WebObjects application by distributing the processing load among multiple instances of the application. These application processes can be running on the same machine as the server or on remote machines. The task that accomplishes this distribution is called *load balancing*.

As an example of how load balancing works, suppose you have an application called MyApp and you have configured WebObjects to run two instances of MyApp on the host toga and two instances on the host tutu. When a user types this URL:

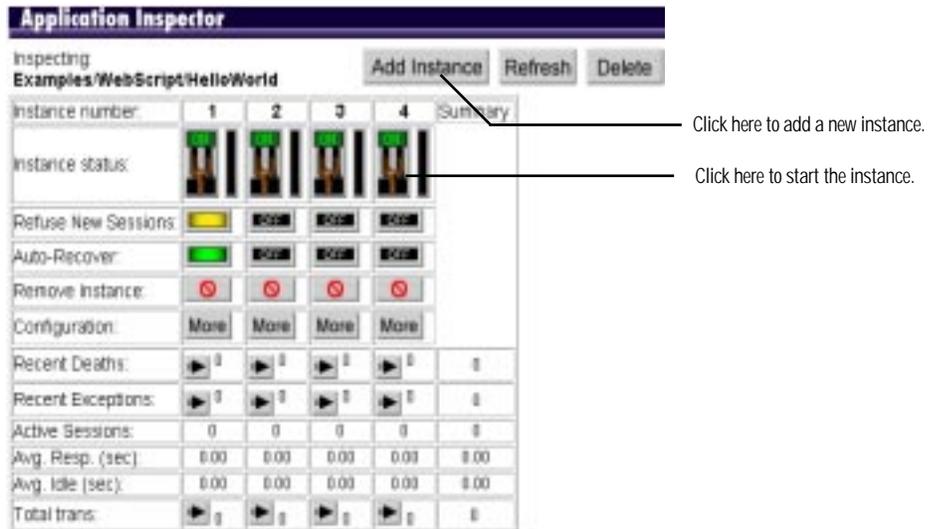
```
http://toga.acme.com/cgi-bin/WebObjects/MyApp
```

the WebObjects adaptor looks for an instance of MyApp on the host toga. If it finds an instance and the instance is ready to receive requests, the adaptor sends the request to that instance. If both of the instances of MyApp on toga are busy, it accesses an instance on the host tutu.

Use the Monitor application to create new instances of an application for load balancing. To create new instances of an application, do the following:

1. Locate the application in the left frame and click the inspector button to its left.

The Application Inspector opens in the right frame.



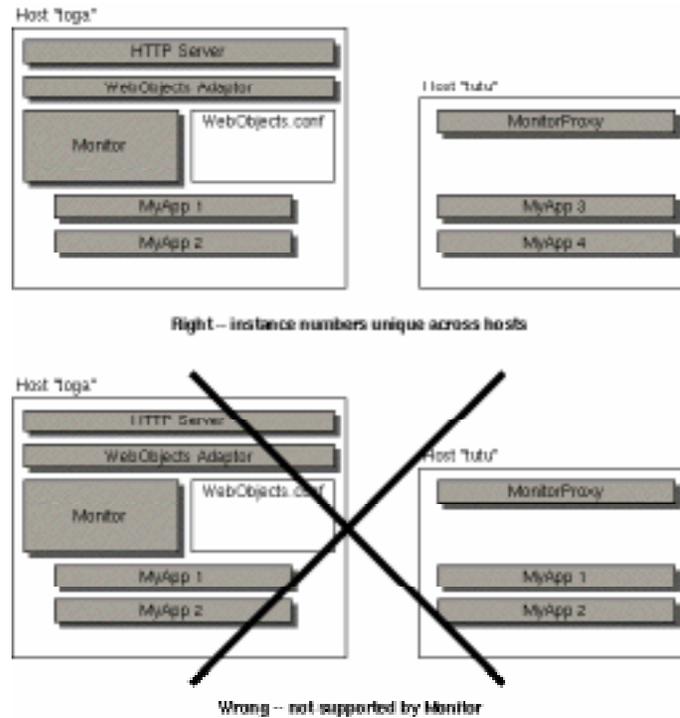
2. Click the Add Instance button to add a new instance of the application.
3. If you want the instance to run on a different host, click the More button for that instance, scroll down in the bottom frame until you see the host name field, enter the name of the host you want that instance to run on, and click the Save Settings button at the bottom of the frame.
4. To start the application instance, click the On/Off switch in the Instance Status field.

When you create multiple application instances in this manner, you are creating the public configuration file `NeXT_ROOT/NextLibrary/WOAdaptors/Configuration/WebObjects.conf`. When the adaptor receives an HTTP request for an application, it first (in its initial mode) checks `WebObjects.conf` for an application instance that is accepting connections and forwards the request to it. The section “WebObjects HTTP Adaptors” describes in some detail both the public configuration file and the adaptor modes involved in load balancing.

Note that Monitor always assigns a unique number to each application instance, even if it is running on a different host. It does this so that it can recover a crashed instance for you. If an instance dies, Monitor can try to recover it by launching it on another host. Because of this, instance numbers must be unique across hosts.

The `WebObjects.conf` file, however, only requires an instance number to be unique on a given host. Consider the example given previously, where two

instances of MyApp run on host toga and two instances run on host tutu. If you were to set up a `WebObjects.conf` file by hand, you could assign instance numbers 1 and 2 to the two instances on toga and instance numbers 1 and 2 to the instances on tutu. This is legal, but it's not supported by the Monitor, and if you do this, you won't be able to use Monitor for the instances you've created.



To determine how many instances of an application you should run, do the following:

1. Test the application using the recording and playback performance tools as described in the section "Performance Testing."
2. Check the application's response times using the Application Inspector in the Monitor application.
3. If the response time is slow, use Monitor to add another instance of the application.
4. Continue to add instances and check their response times. When all instances have reasonable response times, you have the number of instances you need.

Your application's state-storage strategy affects load balancing. By default, applications store state in memory in the server. If the application uses this default state-storage strategy, the instance that processed a session's first request must be used to process all subsequent requests. That is, the load-balancing granularity is per session. If you store state using some other strategy (for example, if you store state in the file system), true load-balancing can be achieved; each request from a session can be processed by any application instance (that is, the load-balancing granularity is per request).

Increasing the Listen Queue Depth

When an application's request load is sometimes heavier than other times, you can increase the listen queue depth to improve performance.

For example, suppose an application can process one transaction per second and it typically receives transactions at the rate of one transaction every two seconds. The application's listen queue remains empty because it can handle the load. Suppose that at certain times of the day, this same application receives a much heavier load of two requests per second. At these times, the listen queue fills up because the application cannot process as many requests as it receives. If you know that the request rate will eventually return to the normal load of one request every two seconds, increasing the listen queue depth will help improve performance during the heavy load time.

On the other hand, suppose that two requests per second becomes the normal request load for this application. In this case, no matter how big the listen queue, the application can never catch up because it only processes one request per second. In this situation, when the average load is higher than the application can handle, load balancing is the proper solution.

To increase the listen queue depth, do the following in Monitor:

1. In the Application Inspector, click the More button to show the Application Instance Inspector in the bottom frame.
2. Scroll down until you see the check box labeled Listen queue depth. Click this check box.
3. Enter the size of the listen queue depth in the field next to the check box.
4. Click the Save Settings button at the bottom of the frame.

Installing and Configuring NSAPI Adaptors

If you have a Netscape server, use one of the NSAPI adaptors. Which NSAPI adaptor to use, and the procedure for configuring it, depends on the type of server you have. Adaptors are located in `NextLibrary/WOAdaptors/NSAPI`.

If you have server...	use adaptor...
Netscape 2.0 (FastTrack/Enterprise)	2.0/WebObjects-NSAPI.[dll so]
Netscape 2.0.1 (FastTrack/Enterprise)	2.0.1/WebObjects-NSAPI.[dll so]
Netscape 3.0 (FastTrack/Enterprise)	3.0/WebObjects-NSAPI.[dll so]

Note: There is no requirement for installing an adaptor anywhere other than its original location. If you wish, you can copy the adaptor to the server's executable or configuration directories, but ensure that the configuration specifications refer to its proper location. The following procedures assume the original installed locations.

To configure Netscape 2.0, 2.0.1, or 3.0 NSAPI adaptors for all platforms, complete the following procedure:

1. Locate the server configuration file in the directory `cgi-bin/config/obj.conf` where `cgi-bin` is the server's cgi-bin directory.
2. Edit the configuration file to insert one an line similar to one of the following:

In the `obj.conf` file insert the following:

```
Init fn=load-modules shlib=c:/NeXT/NextLibrary/WOAdap-
tors/NSAPI/2.0/WebObjects-NSAPI.dll funcs="WONetscapeInterface,WONSIn-
terfaceFindWebObjects"
```

This example is specific to Windows NT and NSAPI 2.0; for Solaris and HP-UX the name of the adaptor executable is `WebObjects-NSAPI.so`.

3. Locate the following line in `obj.conf`:

```
NameTrans from="/cgi-bin" fn="pfx2dir" dir="cgi_bin_dir" name="cgi"
```

Just before this line, insert the following line:

```
NameTrans from="/cgi-bin/WebObjects" fn="WONSInterfaceFindWebObjects"
name="webobjects"
```

4. At the end of `obj.conf`, add the following text, just as it appears here:

```
<Object name="webobjects">  
Service fn="WONetscapeInterface"  
</Object>
```

5. Restart your server.

Notes

On Windows NT, you can restart your server from the Services control panel by stopping and then starting it (clicking the Stop button, then clicking the Start button). However, it is better to use the browser interface provided for administration to restart the server. If there are errors, you can check the error activity log to find out what they are.

When you test an API-based adaptor to verify that it's properly configured, you should eliminate the CGI adaptor as a factor. To do this, rename `WebObjects` (or `WebObjects.exe`) to something like "WebObjects_test" (or "WebObjects_test.exe") and test the API-based adaptor. If you wish later to restore the CGI adaptor, simply undo the changes you made previously.

Installing and Configuring the ISAPI Adaptor

If you have one of Microsoft's Internet Information Servers (IIS), such as the Peer Web server that comes with the NT Workstation or the IIS server that comes with NT Server 4.0, you need to install and configure the ISAPI adaptor that comes with `WebObjects Enterprise`.

Note: This procedure is applicable only to the `WebObjects Enterprise` product on Windows NT platforms.

1. Copy the ISAPI adaptor from its installation location to the server's "Scripts" directory:

```
cp C:/NeXT/NextLibrary/WOAdaptors/ISAPI/WebObjects-ISAPI.dll  
C:/INETPUB/Scripts
```

This example assumes that `NeXT_ROOT` is `C:\NeXT\` and that the IIS server is installed in `C:\INETPUB`. These directories could be different on your system (for instance, the server could be installed in `D:\INETPUB`). This example also shows a copy operation using the `cp` command in a Bourne shell; alternatively, you could copy the DLL using the NT Explorer program or through similar programs.

2. Set up your site so that the proper URL for the ISAPI adaptor is submitted when users click buttons, images, or hyperlinks that have as targets WebObjects applications. This URL has the form:

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

For HTTP requests that use the CGI adaptor, make sure that the URLs conform to this format:

`http://host/Scripts/WebObjects.exe/ApplicationPath`

Starting Up Multiple Monitor Instances

As a fail-safe measure, you can start up multiple instances of the same Monitor application. When you start an instance of Monitor, the instance searches to see if another instance is running. If so, it puts itself to sleep. If the first Monitor instance crashes, the second instance will wake itself. Monitor stores its configuration information and its current state in the file system. Thus, in the event of a crash, the backup instance can take over, read the configuration file, restore the state from the file system, and continue as if nothing had happened.

You can have as many instances of Monitor running as you like. They will order themselves to take over in the event that the controlling instance fails.

The same holds true for `MonitorProxy` daemon. You can have as many of these running as you like.

Remember that Monitor is a WebObjects application. To start multiple instances, you must provide unique instances numbers and port numbers to the adaptor on the command line, as well as the application path argument, like this:

```
> Monitor.exe -a WODefaultAdaptor -n 1 -p 1067 Monitor
> Monitor.exe -a WODefaultAdaptor -n 2 -p 1068 Monitor
```

In this example, the first instance becomes the controlling instance of Monitor. The second instance becomes active only if the first instance crashes.