Serving WebObjects

To a large extent, WebObjects needs little attention once it is installed. However, administrators of a WebObjects site still need to know how to accomplish certain tasks, such as installing applications, creating and running instances of them, and configuring HTTP adaptors. In addition, you'll probably be concerned about improving your site's performance. The tools and techniques described in this document help administrators complete the tasks required to deploy and maintain WebObjects applications. Because each deployment can be different, the document gives suggestions and options for making your deployment successful.

This document begins by providing essential background information on WebObjects HTTP adaptors and how they are used to distribute requests. Then it describes how to use an application called Monitor to monitor and administer your deployment. Finally, it describes the basic administrative tasks are and tells you how to perform them.

Table of Contents

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.

- You can find installation instructions for supported HTTP adaptors in *NEXT_ROOT*/**Library/WebObjects/Adaptors/InstallationInstructions.html**. Instructions for building HTTP adaptors from provided source code are located in *NEXT_ROOT*/**Developer/Examples/WebObjects/Source/Adaptors/BuildingInstructions.html**

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** or **scripts** 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, WebObjects adaptors based on APIs specific to particular web server. The NSAPI adaptor, which is based on the Netscape Server 3.5 API, is available on all supported platforms except the Mach-based Mac OS X Server. A WebObjects adaptor based on Microsoft's Internet Information Server API (ISAPI) is also supported on Windows NT. WebObjects also supports an adaptor based on Apache's module API on UNIX platforms (including the Mac OS X Server). In addition, Netscape's WAI API is provided in this release as an example project but is not supported; the WAI adaptor is suitable for all platforms except Mac OS X Server.

  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.

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*/**Library/WebObjects/Configuration/WebObjects.conf**. (*NEXT_ROOT* is defined at system installation time on Windows NT systems; on Mac OS X Server and similar systems, it is always **/System**.) 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 "Setting Up the Monitor Application" 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 if you a using an API adaptor, or in each server's **cgi-bin** or **scripts** directory if you are using the CGI adaptor.

- A *private configuration file* is also named **WebObjects.conf** and is located in the temporary directory of the system (**/tmp** for Mac OS X Server, 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, it searches the private configuration file. Thus the public configuration file ensures security in deployment mode because only the applications you list are accessible.

  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. This file should only be directly modified by the WebObjects adaptor itself.

Adaptor Modes

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

1. Load-balancing between concurrent instances of the same application specified in the public configuration file (deployment)
2. Choosing an application from the private configuration file (development)

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 or if the adaptor cannot find a matching entry, it returns a page listing the contents of the configuration file. For example, if an application has been stopped, the adaptor might still list its entry in the private configuration file.

Note that if the public configuration file *NEXT_ROOT*/**Library/WebObjects/Configuration/WebObjects.conf** exists, no applications listed in the private configuration file are ever contacted. Also note that adding applications to the Monitor as described in the section "Adding and Configuring an Application" creates the public configuration file. Thus, if you are using the Monitor application, autostarting is disabled.

Installable HTTP Adaptors

When WebObjects is installed, the adaptors listed in the table below, when appropriate to the platform, are put in *NEXT_ROOT*/**Library/WebObjects/Adaptors**; source code for all adaptors is written to *NEXT_ROOT*/**Developer/Examples/WebObjects/Source/WOAdaptors**. Note that only the CGI and Apache adaptors can be found on Mac OS X Server, since neither Netscape's nor Microsoft's servers have been ported to this platform. Also, no ISAPI binary file is written to Solaris or HP-UX platforms (only source code).

The following table summarizes the adaptors provided with WebObjects.

	Server	/Library/WebObjects/ Location	Executable
CGI	many	Adaptors/CGI	WebObjects[.exe]
NSAPI	Netscape 3.51 FastTrack (httpd) Enterprise (https)	Adaptors/NSAPI	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)	Adaptors/ISAPI	WebObjects-ISAPI.dll
Apache	1.3	Adaptors/Apache	mod_WebObjects.o
WAI	Netscape 3.5 servers	Adaptors/WAI	(must build example project)

Note: You can find installation instructions for supported HTTP adaptors in *NEXT_ROOT*/**Library/WebObjects/Adaptors/InstallationInstructions.html**. The procedure for building HTTP adaptors from provided source code is located in *NEXT_ROOT*/**Developer/Examples/WebObjects/Source/Adaptors/BuildingInstructions.html**

Deploying With the Monitor Application

Monitor is an application that facilitates the administration of local and remote deployments of WebObjects applications. Itself a WebObjects application, Monitor provides a simple graphical user interface for performing common administrative tasks such as:

- Adding and removing instances of applications
- Starting and stopping the execution of application instances
- Automatically restarting an instance upon failure
- Sending electronic mail to administrators when an instance fails
- Scheduling instances to be automatically started and stopped at specified intervals
- Configuring instances to be run on remote hosts

Monitor's interface reflects three distinct configurable entities: applications, instances, and hosts. An "application" represents a WebObjects application abstractly. An "instance" represents a specific instance of an application on a particular host; an instance is either running or stopped. A "host" represents a server available to run instances of WebObjects applications.

Setting Up the Monitor Application

When WebObjects is installed, the Monitor application (**Monitor.woa**) is put in *NEXT_ROOT*/**Library/WebObjects/Applications/**. Monitor's images should also be installed in your web server's document root under *DOC_ROOT*/**WebObjects/Monitor.woa**. Verify that both these paths exist before you attempt to start Monitor.

Depending on your platform and on other factors related to your deployment, you might want to configure your server to launch Monitor automatically when it starts up. If you're using Monitor on Windows NT, then see "Setting up Monitor and MonitorProxy as Services on Windows NT" (page 13), which describes how to set up Monitor as a service.

Starting Up Monitor

To start up Monitor, do the following:

1. Open a command shell window.

Use **Terminal.app** on Mac OS X Server and the Bourne Shell program on Windows NT.

2. If you are on a Mac OS X Server (or similar) system, **su** to root.

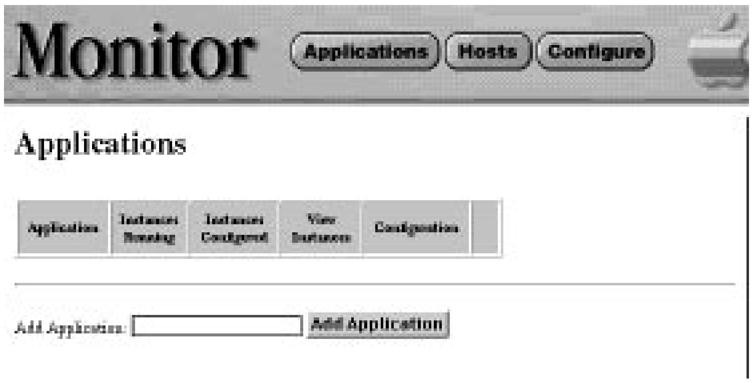
3. Change directories to the location where WebObjects is installed (such as **/System** on Mac OS X Server and **C:\Apple** on Windows NT).

4. Enter the following commands:

```
cd Library/WebObjects/Applications/Monitor.woa
Monitor
```

If you start Monitor on Windows NT platforms by double-clicking the icon in the Windows Explorer program, any application instances started with that Monitor are terminated when the Monitor instance itself terminates (which usually occurs when the Command-Prompt window is closed or upon failure).

When the Monitor application launches, it usually opens the default web browser and displays the Applications Page by default:

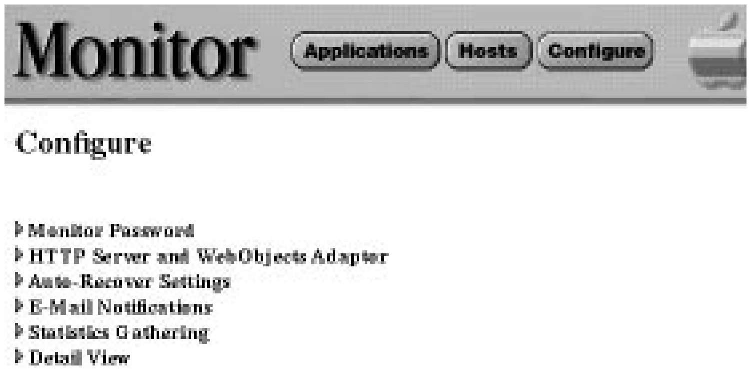


Application	Instances Running	Instances Configured	View Instances	Configuration

Setting Up Monitor

Your first probable task as administrator is to verify and change the configuration settings that Monitor chooses by default, particularly the URL used to locate the adaptor. To do this, complete the following steps:

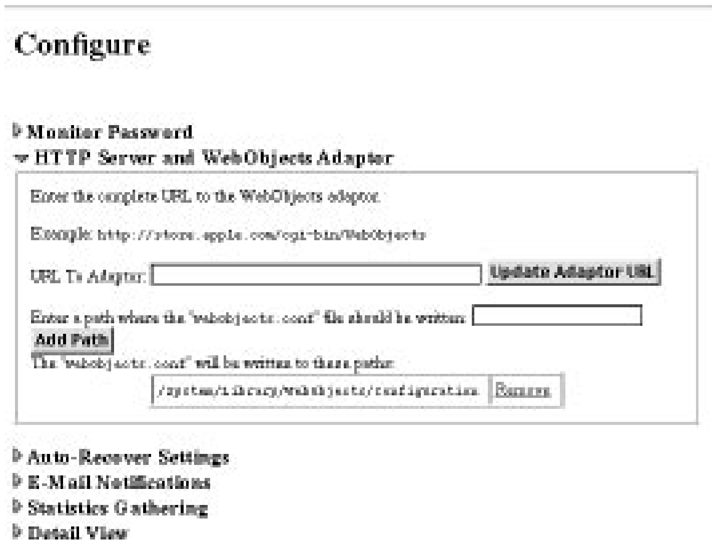
1. Click the Configure button. This brings up the Global Configuration page:



Configure

- Monitor Password
- HTTP Server and WebObjects Adaptor
- Auto-Recover Settings
- E-Mail Notifications
- Statistics Gathering
- Detail View

2. Click the small triangle next to the “HTTP Server and WebObjects Adaptor” item. Doing so causes the display of the following page:



Configure

- Monitor Password
- HTTP Server and WebObjects Adaptor

Enter the complete URL to the WebObjects adaptor.

Example: `http://store.apple.com/cgi-bin/WebObjects`

URL To Adaptor:

Enter a path where the "webobjects.conf" file should be written:

The "webobjects.conf" will be written to these paths:

`/system/library/webobjects/configuration`

- Auto-Recover Settings
- E-Mail Notifications
- Statistics Gathering
- Detail View

3. In the URL To Adaptor field enter the adaptor’s URL

   This URL should include the web server from which clients will access applications plus the remaining portion of the URL up to the WebObjects adaptor.

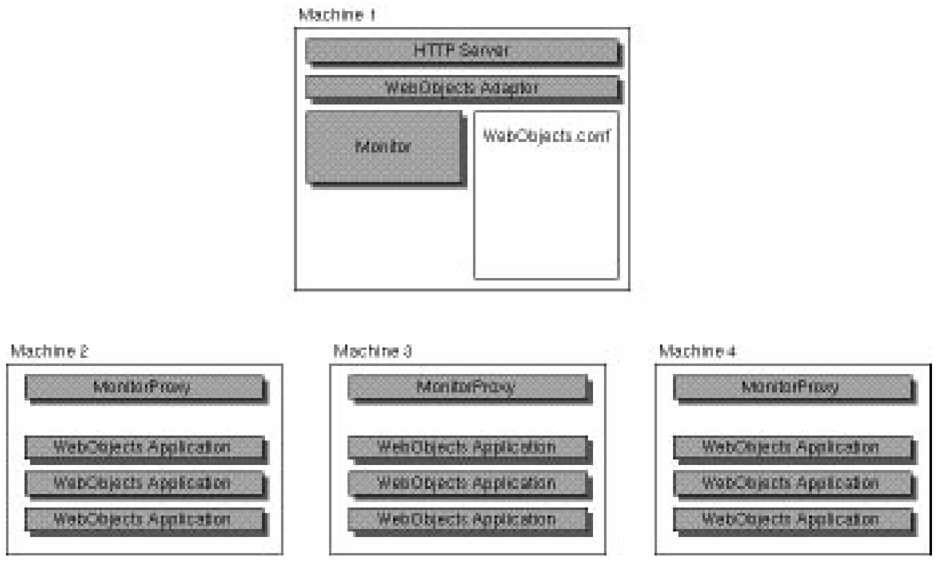
4. Click Update Adaptor URL.

Setting the adaptor URL is the minimal setup task required for administering applications on the local machine. You might want to fine-tune your site’s configuration and take advantage of other features such as e-mail notifications.

Deploying on Multiple Hosts

Creating a deployment environment sometimes involves more than one HTTP server and many WebObjects application instances running on each server. The Monitor application is designed to run on a single machine. Thus there can only be one copy of Monitor running at a time and managing the same set of hosts. To make several hosts available to Monitor, a service called **MonitorProxy** must be running. With a **MonitorProxy** running the Monitor application can remotely administer a host machine.

A large WebObjects deployment could be depicted as in the following diagram:

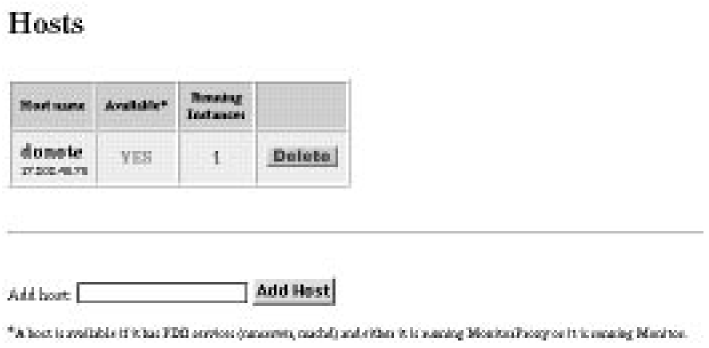


Machine 1 acts as the web server and load balancer between all the application servers running on Machines 2, 3, and 4. You should run Monitor on the same host as the web server and the WebObjects adaptor since Monitor is involved in modifying and updating the **WebObjects.conf** file. which the adaptor uses to find instances.

Adding a Host to Monitor

Before you run **MonitorProxy** on a remote machine, you should let the Monitor application know about it. To do this, complete the following steps:

1. Click the Hosts button in the Monitor banner. The following page is displayed:



Hosts

Host name	Available*	Running Instances	
donate [illegible]	YES	1	Delete

Add host: Add Host

*A host is available if it has [illegible] services ([illegible]) and either it is running MonitorProxy or it is running Monitor.

2. Enter the name of a host in the “Add host” field. It must be a valid host name assigned to an IP address (that is, it must have a DNS entry).

3. Click Add Host.

Running MonitorProxy

The **MonitorProxy** executable can be found inside the **Monitor.woa** application directory in *NEXT_ROOT*/**Library/WebObjects/Applications.** To start the **MonitorProxy** service on a remote machine:

1. Open a command shell (using, for instance **Terminal.app** on Mac OS X Server or the Bourne Shell program on Window NT).

2. Enter “MonitorProxy” with no arguments to start the service.

MonitorProxy launches and searches for a Monitor application running on a nearby host. **MonitorProxy** might not find a host running Monitor, but do not be alarmed. It’s more important that Monitor finds the **MonitorProxy** than vice versa. Go to the Hosts page in Monitor to see if Monitor detects that the host is available and running a **MonitorProxy.**

You might find that **MonitorProxy** takes a long time to start up because it is searching for the Monitor’s host. To resolve this problem, specify the host running the Monitor as a command-line argument when you run **MonitorProxy.** For example:

```
MonitorProxy –mhost server1
```

You might want to set up your system so that **MonitorProxy** starts up at system boot time. See “Setting up Monitor and MonitorProxy as Services on Windows NT,” below, for instructions on doing this on Windows NT.

Setting up Monitor and MonitorProxy as Services on Windows NT

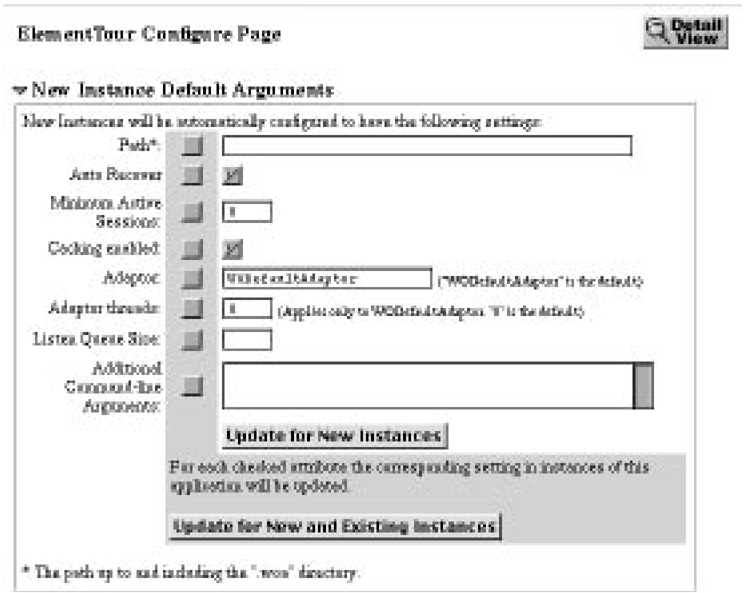
If you are using Windows NT as your deployment platform then you may find it useful to start Monitor and **MonitorProxy** as services. When the operating system starts up, it then starts Monitor and **MonitorProxy** automatically.

When WebObjects is installed on your NT machine, some services are created. To configure these services, open the Control Panel and click the Services Icon. Look for a service named "Apple WebObjects Monitor" and another named "Apple WebObjects MonitorProxy." You can configure these services to start up automatically.

Adding and Configuring an Application

To add a new application to Monitor, click the Application button in the banner. This brings you to the Applications page, which lists all configured applications. In the Add Application text field, enter the name of a new application; this should be the same name as the application project, or the wrapper name minus the **.woa**. For the ElementTour example application, the entered string would be "ElementTour".

When you enter the name of your new application and click the Add Application button, Monitor displays the Application Configuration page:



ElementTour Configure Page

New Instance Default Arguments

New Instances will be automatically configured to have the following settings:

Path*:		
Auto Recover:		✓
Minimum Active Sessions:		0
Caching enabled:		✓
Adaptor:		WODefaultAdaptor ("WODefaultAdaptor" is the default)
Adaptor threads:		8 (Applies only to WODefaultAdaptor. "8" is the default)
Listen Queue Size:		
Additional Command-line Arguments:		

For each checked attribute the corresponding setting in instances of this application will be updated.

* The path up to and including the ".woa" directory.

First enter the full path to the WebObjects application in the Path field. For ElementTour running on Windows NT you might enter a string similar to the following example:

```
C:/Apple/Developer/Examples/WebObjects/Java/ElementTour/ElementTour.woa
```

Be sure that the path specifies the built WebObjects application, including the **.woa** extension. You cannot start an instance of an application when the wrong path is specified, and Monitor will not provide feedback when you attempt to start such an instance. The executable name must be the name of the application—in this case "ElementTour" (or, on Windows NT, "ElementTour.exe"). Click the Update for New Instances button on the bottom of the form to save your changes.

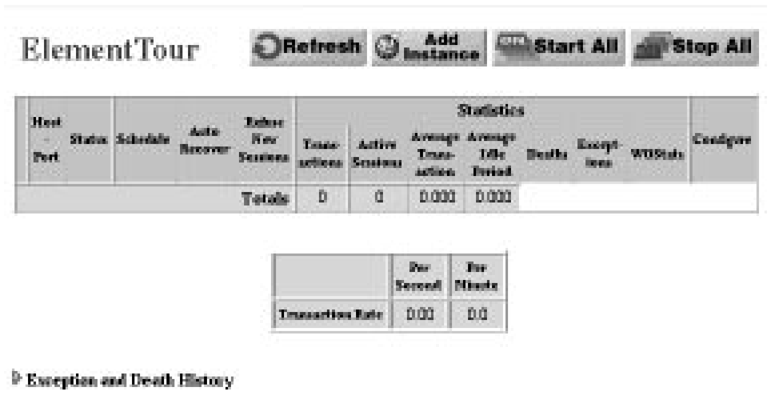
The other fields on this form accept arguments to use when the application instance is run. For descriptions of these fields and as well as the checkboxes and the Update for New and Existing Instances button, see "Setting Command-Line Arguments in Monitor" on page 24.

Creating Application Instances

Each application instance you create adopts the defaults provided in the Application Configuration page for the application. To create an instance,

1. Click the button labeled "Detail View" in the upper right corner of the Application Configuration page.

   The application's Detail View page is then displayed:

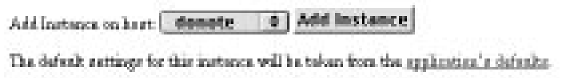


Host - Port	Status	Schedule	Auto Recover	Refuse New Sessions	Statistics							Configure
					Transactions	Active Sessions	Average Transaction	Average Idle Period	Deaths	Exceptions	WOStats	
				Totals	0	0	0.000	0.000				

	Per Second	Per Minute
Transaction Rate	0.00	0.0

2. Click the Add Instance button to create a new instance of your application.

   A new page appears that gives you a choice of hosts to add your instance to.



Add Instance on host: donate

The default settings for this instance will be taken from the application's defaults.

3. Select the host from the pop-up menu.

   Unless you previously configured hosts in Monitor, there should only be one item in the pop-up menu.

4. Click the Add Instance button.

After clicking this button you are returned to the Detail View page, where you can now see a new row in the table showing the status of the instance you just created. From this page you can start the application instance. See "Starting and Stopping an Application Instance on page 20" for details.

When you use Monitor to add an application or an instance, or indeed to change any setting, you are creating or updating the public configuration file *NEXT_ROOT*/**Library/WebObjects/Configuration/WebObjects.conf**. The presence of this file tells the adaptor to do load balancing across multiple instances. See "Configuration Files" on page 5 for more information on **WebObjects.conf**.

Monitor Option Reference

Many of Monitor's options are listed on pages with triangles to the left of them. Clicking the triangle causes a section (a form with fields, buttons, and so on) to be displayed.

Global Configuration

The Global Configuration page allows you to configure aspects of Monitor and your site that are not specific to an application. It lists the options described below.

To Access: Click the Configure button in the Monitor banner.

Monitor Password

In this section specify a password that is required to access Monitor. Monitor does not have a password set by default. After you set a password Monitor prompts users for it each time they access the application.

HTTP Server and WebObjects Adaptor

You use this section primarily to specify the URL that points to the adaptor on your deployment's web server. Monitor uses this URL to construct more URLs that direct the administrator to running instances, the WOStats page, and other destinations.

Also configurable is the paths to which Monitor writes new versions of the **WebObjects.conf** file used by the adaptor for load balancing. You can specify multiple locations for Monitor to write this file in case you have a customized adaptor or if your web server is on a different machine and you need to have the file written to a network-mounted drive.

Auto-Recover Settings

This configuration option allows you to have an additional level of recovery outside the per-instance Auto-Recover feature. This feature restarts application instances after they fail. When Monitor is notified that an instance has crashed, it checks the instance's auto-recover setting to see if it should start a new instance. With the global auto-recover option, when Monitor is started (perhaps when a machine is booted) it can locate all instances configured to be auto-recovered and start them if they are not running. This allows you to have a machine that has failed to boot up, start Monitor, and then have Monitor start all the appropriate applications.

When this setting is on, Monitor perform this check on regular intervals. If Monitor fails or you change an instance's Auto-Recover setting, Monitor launches the appropriate instances within 45 seconds.

E-Mail Notification

In this section you specify an SMTP server that Monitor uses to send e-mail to a set of addresses when application instances fail unexpectedly. In order for Monitor to send e-mail it requires the name or IP Address of an SMTP server.

Statistics Gathering

Periodically Monitor gathers statistical information from each running instance of an application and uses it to refresh the Detail View. If you have many instances of an application, collecting this information on every refresh of the Detail View can be too time consuming. This section allows you to set how long Monitor waits before refetching the statistical information from each instance. The more instances you have the higher this number should probably be. The fewer instances you have the lower this number can be.

Detail View

In this section you can set the interval at which the Detail View page is automatically refreshed.

Host Configuration

This page displays the hosts that Monitor is currently aware of, the status of each host, and whether that host currently can run instances of WebObjects applications. It also displays the number of instances currently running on each host.

To Access: Click the Hosts button in the Monitor banner.

A host machine can run WebObjects applications if it has the WebObjects Deployment packages installed and is running the MonitorProxy service. In order to run an instance on a host remote from Monitor, you must add the host in this section (see "Deploying on Multiple Hosts" on page 11). You should use alphanumeric DNS names to refer to hosts. If you enter the name of a host that does not exist, it can take Monitor up to 30 seconds to respond that it cannot contact that host.

Application Configuration Options

The Application Configuration page allows you to configure general aspects of an application. It lists the options described below.

To Access: Click the Applications button on the Monitor banner, then click the Config button next to any instance.

New Instance Default Arguments

In this section you can specify a set of default arguments that Monitor uses when it creates new instances of the application. Most of these arguments are the normal command-line options used to configure an instance of a WebObjects application (see "Starting Up Applications From the Command Line" for descriptions of these options). Two options, Auto Recover and Minimum Active Sessions, are not command-line options but are deployment settings that Monitor uses for determining starting and stopping policy.

The "Setting Command-Line Arguments in Monitor" of this document discusses how to set launch options using Monitor.

These settings all appear in the Instance Configuration page as well.

Scheduling Instances

This section enables the administrator to configure an application's pool of instances to conform to a staggered schedule of starting, running for a period of time, begin refusing new sessions, and shutting down when the minimum active session threshold is reached.

See “Automatic Scheduling” on page 37 for instructions on setting the shutdown and startup schedules for all instances of an application as well as specific instances.

E-Mail Notifications

If the SMTP server has been set in Monitor’s Global Configuration page (see “Global Configuration” on page 15) then an application can specify a list of electronic-mail addresses to send an mail to when an instance fails unexpectedly. This list should be comma delimited (for example, “jdoe@somewhere.com, mpublic@else.com, foo@bar.com”).

The mail message contains the application name, host, port, and date and time of the failure.

To turn off the feature, delete all addresses from the text field and click Update.

Instance Configuration Options

With the options of the Instance Configuration page you can override global application settings for particular instances.

To Access: From the Detail View for an application, click the Config button next to the desired instance.

Application Start-Up / Command Line Arguments

This section allows you to change the command-line arguments that are used when the instance is started. See “Setting Command-Line Arguments in Monitor” on page 24 for details.

For convenience, the entire set of command-line arguments passed to the instance are displayed in the blue box at the bottom of this section.

Graceful Shutdown

This section allows you to change the minimum active session threshold for an instance. This threshold is used when the instance begins refusing new sessions. The default is zero. If your application is usually under heavy traffic, you might not want to wait for all sessions to time-out before terminating the application.

Scheduling

This section allows you to configure the scheduling settings for a given instance. Normally you should use the Application level scheduling to create a staggered schedule of starting and stopping instances. Use the instance-specific section to create your own schedule intervals. See "Automatic Scheduling" on page 37 for the scheduling procedure.

Monitor computes from the desired instance lifespan or from the desired instance downtime a series of shutdown dates The scheduling algorithm causes the instance to begin refusing new sessions on regular intervals based on these two variables.

Administrative Tasks

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

Installing Applications
Starting and Stopping an Application Instance
Monitoring Application Activity
Performance Testing
Improving Performance
Automatic Scheduling
Load Balancing
Increasing the Listen Queue Depth
Making Monitor and MonitorProxy Fail-safe

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)/Library/WebObjects/Applications`.

   If you're installing a framework, type `$(NEXT_ROOT)/Library/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:

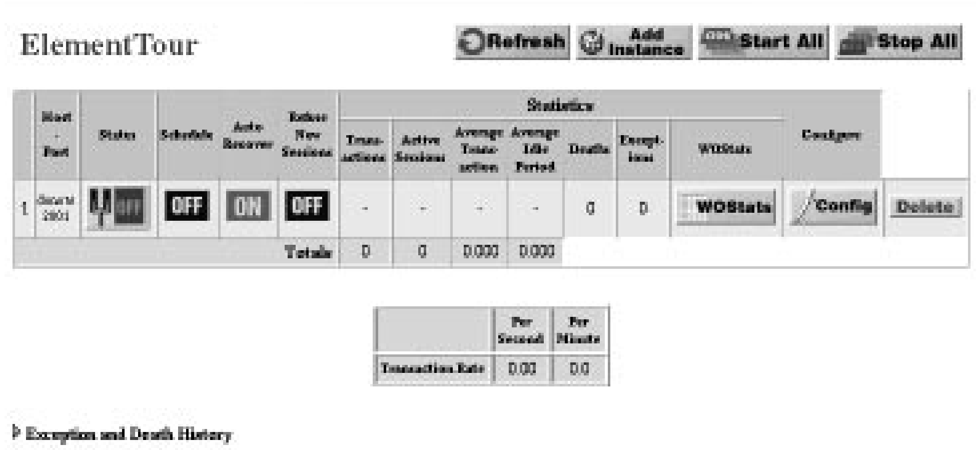
```
NEXT_ROOT/Library/WebObjects/MyApp.woa
  MyApp[.exe]
  Resources/
  WebServerResources/

<DocRoot>/WebObjects/MyApp.woa
  WebServerResources/
```

As discussed 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*/**Library/WebObjects/Applications**. 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*/**Library/WebObjects/Applications** 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's base URL (WOApplicationBaseURL) on the command line. For more information, see "Starting Up Applications From the Command Line" in this guide.

Starting and Stopping an Application Instance

You start an instance from the Application Detail View page. To get to this page, click Applications in the top banner, then click the Detail View button in the row of the instance you wish to start. If you have just added an application instance, and are in the Application Configuration page for the application, click the Detail View button at the top right of the page. A page similar to the following should then appear:



	Host - Port	Status	Schedule	Auto Recover	Refuse New Sessions	Statistics						WOStats	Configure	
						Transactions	Active Sessions	Average Transaction	Average Idle Period	Deaths	Exceptions			
1	[illegible] 2001	OFF	OFF	ON	OFF	-	-	-	-	0	0	WOStats	Config	Delete
					Totals	0	0	0.000	0.000					

	Per Second	Per Minute
Transaction Rate	0.00	0.0

Exception and Death History

The button that looks like a power switch reports the current state of your instance: ON or OFF. The rest of the table reports other information about your instance (for more details see “Obtaining Information From Monitor” on page 27).

1. Click the power switch.

   The Detail View page is refreshed and the power switch appears in an animated toggle state, signifying that Monitor is trying to start your instance.

2. After a few seconds, click the Refresh button.

   Monitor will refresh the Detail View page and with success your instance will be running and the power switch will be on.

If successful, this procedure starts an instance of the application but does not display it in the web browser. When one or more instances are running, the name of the application above the table of instances turns green, becoming a hyperlink that, when clicked, access an instance of the application. In addition, the host name and port number for each instance also become hyperlinks; clicking one of these accesses a specific instance.

If after completing the startup procedure, the instance’s power switch is off, it might be due to one of the following reasons:

- Your instance failed to start and exited; check the instance’s error messages to find out why.

- Your instance is still starting up and Monitor has not received notification.

- Monitor couldn't start your instance because the path was wrong or the executable did not exist.

Monitor starts an instance of your application by creating a new task with the executable; it passes along all the appropriate arguments from the Instance Configuration page for that instance. Monitor starts instances of your application in one of two ways, depending on whether the application is on a different host.

- If the application instance 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 cannot contact a **MonitorProxy**, it does not start the application.

- If the application instance is to run on the same host as the Monitor, Monitor starts the instance itself.

Clicking the Status switch for an instance when it is ON stops the instance. Clicking the Start All button causes Monitor to attempt to start all application instances that are currently stopped; clicking the Stop All button causes Monitor to stop all instances that are currently running.

Starting Up Applications From the Command Line

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

```
AppExecutable [ -WODebuggingEnabled YES|NO]
[ -WOAutoOpenInBrowser YES|NO]
[ -WOMonitorEnabled YES|NO [ -WOMonitorHost hostname|subnet]]
[ -WOCachingEnabled YES|NO]
[[ -WOAdaptor adaptorClass] [ -WOPort portNumber]
[ -WOListenQueueSize listenQueueSize]]
[ -WOWorkerThreadCount int] [-WOOtherAdaptors plist]
[ -WOCGIAdaptorURL path] [ -WOApplicationBaseURL path]
[ -WOFrameworksBaseURL path] [ -NSProjectSearchPath plist]
[ -WOIncludeCommentsInResponses YES|NO] [-WOSessionTimeout seconds]
```

The *AppExecutable* variable represents 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*/**Library/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*/**Library/Executables**.

The following table describes each command-line option:

Option	Description
-WODebuggingEnabled YESlNO	Sets whether the application prints messages to standard error during startup. By default, this option is enabled. WOApplication, WOComponent, and WOSession define a new **debugWithFormat:** method (**debugString** in Java). This method is similar to **logWithFormat:** except that it only prints messages if the **WODebuggingEnabled** option is on
-WOAutoOpenInBrowser YESlNO	Sets whether the application automatically opens a web browser window to the application's URL (starting up the browser if necessary). By default, this option is enabled.
-WOMonitorEnabled YESlNO	Enables or disables monitoring. By default, this option is disabled. If this option is enabled and you manually start an application, the application tries to find a running WOMonitor.
-WOMonitorHost *hostName* | subnet	If the **WOMonitorEnabled** option is on and you use this option, the application tries to find a running WOMonitor on the machine named *hostName* instead of on the local machine. If **subnet** is used, the application looks for a running WOMonitor in its network subnet.
-WOCachingEnabled YESlNO	Requests that the application cache component definitions (templates) instead of reparsing HTML and declaration files upon each new HTTP request. By default, this option is disabled.
-WOAdaptor *adaptorClass*	The WOAdaptor class name. The default is now WOMultiThreadedAdaptor.
-WOPort *portNumber*	The socket port used to connect to an application instance. Unlike previous versions of WebObjects, this option is independent of the adaptor option. A *portNumber* of -1 means use an arbitrary high port number; however, you cannot specify -1 as the value on the command line; to set the value to -1, you must use the **defaults** command.
-WOListenQueueSize *listenQueueSize*	The depth of the listen queue. The default is now 5, meaning that while the application process is handling a request, up to five 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 necessarily 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.
-WOWorkerThreadCount *int*	Maximum number of worker threads for a multithreaded application. The default worker thread count is 8. Setting this count to 0 results in single-threaded (WebObjects 3.5-style) request dispatch.
-WOOtherAdaptors *plist*	Use this option to attach additional adaptors (other than the one specified by **-WOAdaptor**) to the application. The *plist* option is an array of dictionaries written in property-list format.
-WOCGIAdaptorURL *path*	The absolute URL that points to the WebObjects CGI adaptor.

Option	Description
-WOApplicationBaseURL *aURL*	The URL where your application's resources are located under the web server's document root. You may place your application anywhere under the document root. This option is required when you're using a web server. If you install the application in a subdirectory of ***<DocRoot>*/WebObjects**, you should set this to point to the exact location of the application directory. If you don't set the application's base URL, your application can still run but it cannot find image files and other web server resources.
-WOFrameworksBaseURL *aURL*	The location of frameworks under your document root if you're using a web server. The default is **/WebObjects/Frameworks** (as it was in release 3.5). All frameworks that your application uses must be in this directory.
-NSProjectSearchPath *pList*	An array of paths in which your project directories are located. (The array is written in property-list format.) The default is a single item: ".." If you specify this option, WebObjects looks in the locations you specify for a project that has the same name as the application or framework being loaded. If it finds a project, it uses the images, scripted components, and other resources from the project directory instead of from the application or framework's main bundle. This way, you can modify images and scripted components in your project and test them without having to rebuild the application.
-WOIncludeComments InResponses YES|NO	Sets whether the HTML parser includes comments from the components' HTML files in the responses. The default is YES.
-WOSessionTimeout *timeout*	Sets the timeout interval for sessions. By default, they now time out after 3600 seconds.

You can also set these options programmatically or by using the **defaults** utility. Be careful when setting options programmatically. Most options require knowledge of the environment in which the application runs, and the appropriate values change if you move the application to a different machine. For example, you should never set the **WOPort** option programmatically.

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.

Setting Command-Line Arguments in Monitor

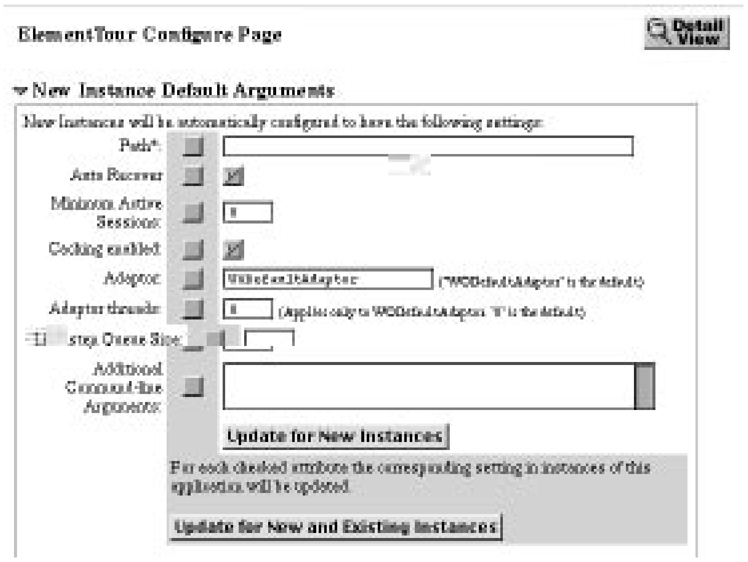
When you use Monitor to start an instance of an application, it uses a set of arguments to initialize that instance. Most of these arguments are the command-line arguments described in “Starting Up Applications From the Command Line” on page 22. You can change an applications arguments, even for all instances that are currently configured and running, by doing the following:

1. Display the Application Configuration page for an application. You can get to this form using one of two approaches:

- Existing applications: Click the Config button next to an instance in the Applications page (which you can get by clicking Applications in the Monitor banner).
- New applications: In the Applications page, enter the name of an application in the Add Application field and click the Add Application button.

2. Click the arrow next to the New Instance Default Arguments option of the Application Configuration page. (This step is not necessary if you are configuring a new application.)

   The following form is exposed:



ElementTour Configure Page

New Instance Default Arguments

New Instances will be automatically configured to have the following settings:

Path*:	☐	
Auto Recover	☐	✓
Minimum Active Sessions:	☐	0
Caching enabled:	☐	✓
Adaptor:	☐	WODefaultAdaptor ("WODefaultAdaptor" is the default)
Adaptor threads:	☐	8 (Applies only to WODefaultAdaptor. "8" is the default)
Li[illegible]sten Queue Size:	☐	
Additional Command-line Arguments:	☐	

For each checked attribute the corresponding setting in instances of this application will be updated.

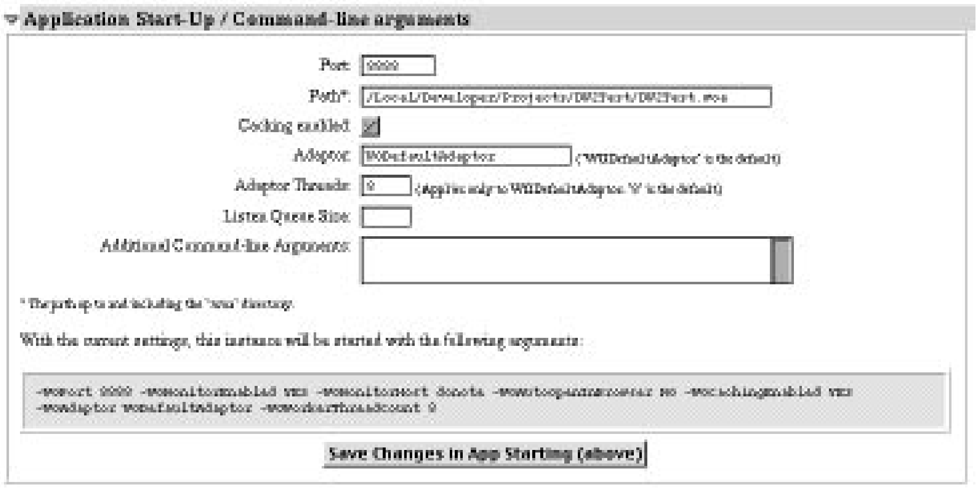
3. Specify the command-line options you want your application's instances to have. For the most common options, simply specify the value (not the key). These options are:

Field	Option
Auto Recover	Internal flag. Specifies whether Monitor should try to restart the instance if the instance fails.
Minimum Active Sessions	Internal flag. Specifies the minimum number of active sessions allowed.
Caching enabled	Command-line option -WOCachingEnabled
Adaptor	Command-line option -WOAdaptor
Adaptor threads	Command-line option -WOWorkerThreadCount
Listen Queue Size	Command-line option -WOListenQueueSize

   For command-line arguments not included in the above table, enter them as "-*key value*" pairs, separated by spaces, in the Additional Command-line Arguments field.

4. If you want the new settings "pushed" to existing instances, click the checkbox in the gray area next to each option, then click the Update for New and Existing Instances button. The existing instances will have to be restarted for new options to take effect.

You can also set the command-arguments for specific instances. To navigate to the form for doing this, go to the Detail View page for an application and click the Config button next to an instance. A list of instance-specific options is displayed; from the list choose "Application Start-Up/Command-line arguments." The following form is exposed:



Application Start-Up / Command-line arguments

Port:	9999
Path*:	/Local/Developer/Projects/DW2Part/DW2Part.woa
Caching enabled:	✓
Adaptor:	WODefaultAdaptor ("WODefaultAdaptor" is the default)
Adaptor Threads:	8 (Applies only to WODefaultAdaptor. "8" is the default)
Listen Queue Size:	
Additional Command-line Arguments:	

* The path up to and including the ".woa" directory.

With the current settings, this instance will be started with the following arguments:

```
-WOPort 9999 -WOMonitorEnabled YES -WOMonitorHost [illegible] -WOAutoOpenInBrowser NO -WOCachingEnabled YES
-WOAdaptor WODefaultAdaptor -WOWorkerThreadCount 8
```

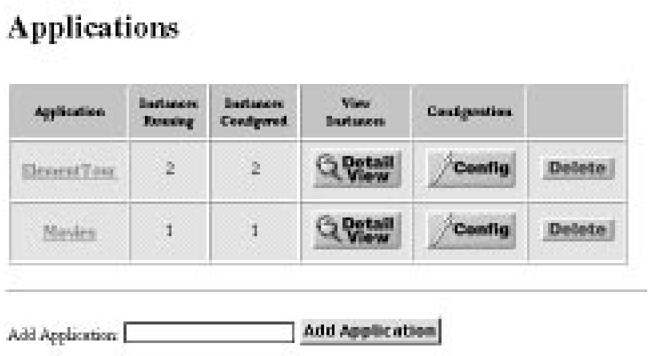
Enter the new arguments and click Save Changes in App Starting. Unlike the previous form, this one allows you to specify a specific port, but it doesn't allow you to set Monitor-specific options (such as auto-recover). Moreover, the changes that you make here do not take effect until the instance is restarted.

Monitoring Application Activity

There are several ways to 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.

Obtaining Information From Monitor

The Applications page gives an overall view of a deployment. It shows which applications are configured, how many instances each application has, and which of these is currently running. You get to the Applications page by clicking the Applications button in Monitor's banner. A screen similar to the following example is displayed:



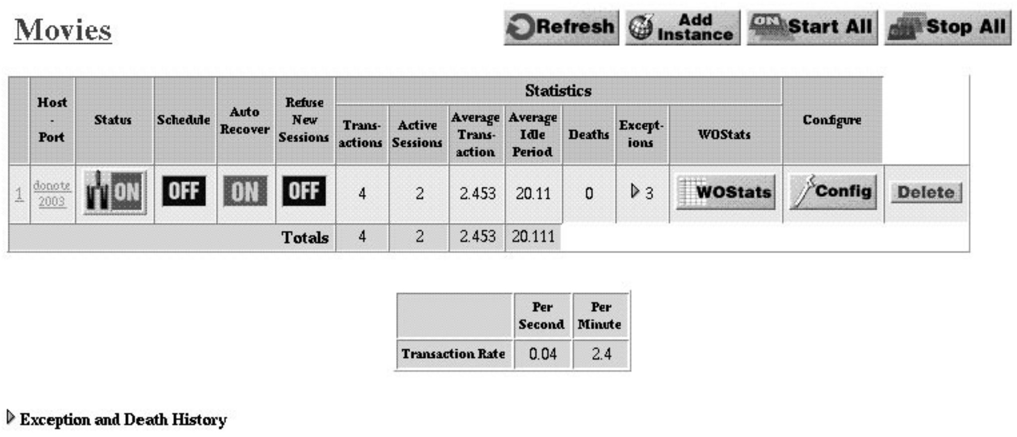
Applications

Application	Instances Running	Instances Configured	View Instances	Configuration	
[illegible]	2	2	Detail View	Config	Delete
Movies	1	1	Detail View	Config	Delete

Add Application: Add Application

Click a hyperlink in the Application column to start a session with an instance of that application.

The Application Detail View page of the Monitor application provides you with detailed information about all configured instances of a WebObjects application. Click the Detail View button next to an application in the Applications page to go to the detail page, which looks similar to the following example:



Movies

	Host - Port	Status	Schedule	Auto Recover	Refuse New Sessions	Statistics							Configure	
						Trans-actions	Active Sessions	Average Trans-action	Average Idle Period	Deaths	Except-ions	WOStats		
1	donote 2003	ON	OFF	ON	OFF	4	2	2.453	20.11	0	▷ 3	WOStats	Config	Delete
					Totals	4	2	2.453	20.111					

	Per Second	Per Minute
Transaction Rate	0.04	2.4

▷ Exception and Death History

At the top of the page is the title of the application. When one or more instances of an application are running then, this title becomes a hyperlink. Clicking on the hyperlink opens a new browser window and connects to the running application.

The tables of the Application Detail View contain various information and controls:

Column	Description
Host / Port	The host name and the port that the instance runs on. If the instance is running, this information is hyperlinked; clicking starts a new session with the instance.
Status	Indicates whether the instance is running (ON) or is stopped (OFF). Clicking this control starts and stops the instance (see “Starting and Stopping an Application Instance” for details).
Schedule	Indicates whether scheduling is enabled. When ON is displayed, the Status, Auto-Recover, and Refuse New Sessions indicators are disabled; scheduling is responsible for setting all of those states on a schedule basis. See “Automatic Scheduling” for information on scheduling.
Auto-Recover	Displays the Auto-Recover setting for this instance. ON indicates that Monitor should start a new instance upon failure or shutdown of an instance. You can set this state when you configure the instance (see “Setting Command-Line Arguments in Monitor”).
Refuse New Sessions	Displays whether the instance is refusing new sessions (YES). If this is the case, all requests from new clients are redirected to another instance that is not refusing.
Statistics.	
Transactions	Total number of requests this instance has serviced since it was started.
Active Sessions	Total number of sessions that are still active for the instance.
Average Idle Period	the average amount of time that the instance is idle between requests.
Deaths	The number of unexpected failures or deaths this instance has had. These exclude “expected” deaths, which include scheduled shutdowns or a manual shutdowns (using Monitor’s interface).
Exceptions	If your instance has an uncaught exception, Monitor may record the number of these exceptions here. When there are exceptions, a small blue triangle appears; click this to inspect the messages describing the exceptions.
WOStats	Click this button to open a new browser window and connect to the WOStats direct action to view detailed statistics about this instance.
Config	Click this button to link to the Instance Configuration page for this instance.
Delete	Click to remove this instance permanently. Deleting an instance also terminates the instance immediately if it is running.
Transaction Rate	This small table indicates the overall transaction rate for the current application. This table reflects the number of transactions that the application as a whole (all of its instances) is servicing per minute and per second.

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; do not use it during deployment.

The temporary directory depends on the platform:

- **/tmp** on Mac OS X Server, Solaris, and HPUX
- 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
   ```

   On UNIX-based systems, you must have root privileges.

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

Most WebObjects applications automatically include a WOStats page and record statistics about themselves in that page while they run. To look at these statistics, access the WOStats page while the application is running. You can do this through Monitor or through any browser.

- In Monitor, go to the Detail View page for an application and click the WOStats button next to an instance.
- From a browser, access the WOStats page with a URL like the following:

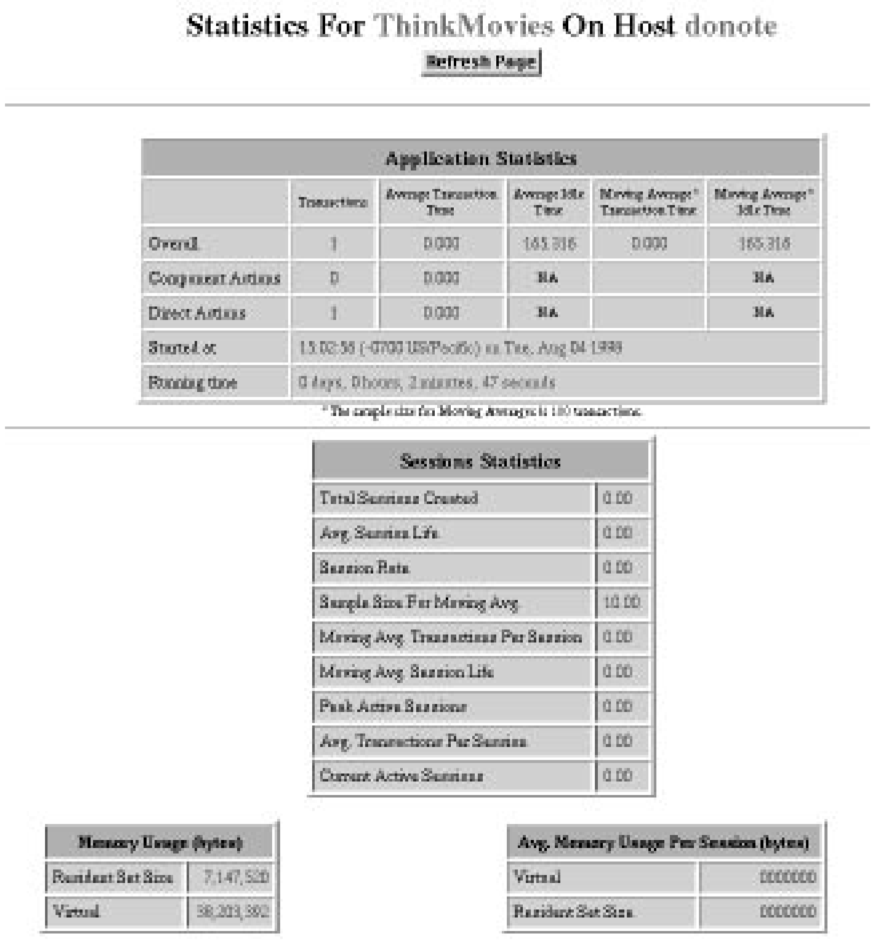
  ```
  http://myhost/cgi-bin/WebObjects/MyWebApp.woa/wa/WOStats
  ```

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

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

  The “1” just before “/wa” is the instance number.

The WOStats page looks similar to the following example:



Statistics For ThinkMovies On Host donote

Application Statistics					
	Transactions	Average Transaction Time	Average Idle Time	Moving Average* Transaction Time	Moving Average* Idle Time
Overall	1	0.000	165.316	0.000	165.316
Component Actions	0	0.000	NA		NA
Direct Actions	1	0.000	NA		NA
Started at	15:02:58 (-0700 US/Pacific) on Tue, Aug 04 1998				
Running time	0 days, 0 hours, 2 minutes, 47 seconds				

* The sample size for Moving Averages is 100 transactions.

Sessions Statistics	
Total Sessions Created	0.00
Avg. Session Life	0.00
Session Rate	0.00
Sample Size For Moving Avg.	10.00
Moving Avg. Transactions Per Session	0.00
Moving Avg. Session Life	0.00
Peak Active Sessions	0.00
Avg. Transactions Per Session	0.00
Current Active Sessions	0.00

Memory Usage (bytes)	
Resident Set Size	7,147,520
Virtual	38,203,392

Avg. Memory Usage Per Session (bytes)	
Virtual	0000000
Resident Set Size	0000000

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

Performance Testing

WebObjects comes with a set of tools that allows you to record a session and then play it back. Using these tools, you can test your application setup to determine whether you have the appropriate number of instances running, the appropriate amount of memory allocated, and so on. The performance tools include:

- The default application adaptor that, when the –**WORecordingPath** flag is set to YES, enables the recording of sessions
- A command-line Java tool that plays back recorded sessions

- A Playback Manager application that can play back sets of sessions (*NEXT_ROOT*/**Library/WebObjects/Applications/PlaybackManager.woa**)

The recording tools are not designed to handle automated functional testing, only performance testing. They simply save requests and play them back after substituting the appropriate session and context identifiers. This means that the playback tool expects the application to return the same page and content and when it was recorded.

This section focuses on recording and playing back sessions from the command line. For information on the Playback Manager application, consult the application's online help.

Recording a Session

When a WebObjects application is launched in recording mode, it saves each request and response made to a recording file (which has an extension of **.rec**). You specify the path designating this file with the **-WORecordingPath** flag, which also serves as a switch to turn on recording. The application automatically appends the **.rec** extension to the given filename and creates a directory, if one doesn't exist, with the given path.

To run an application in recording mode:

1. Start the application on a command line similar to the following:

   ```
   myApplication -WOAutoOpenInBrowser NO -WORecordingPath
   /tmp/TestMyApp/tape1
   ```

   This command creates the file **/tmp/TestMyApp/tape1.rec**.

2. Using a web browser, run a session of your WebObjects application.

   You might want to record what you believe to be a typical session, or you might want to record a session that puts a maximum load on your application. For example, you may want to record a session that performs as many database fetches as possible. As you run the application, the WebObjects recording adaptor writes each request and response to the recording file.

   Keep in mind that all request and responses are saved to disk, so it's recommended that only one user (that is, one session) access the application while recording is underway. You can later play back a recorded session multiple times to simulate more users.

3. Stop the application to stop recording

Playing Back a Session

Once you have recorded a session with your application, you can use the **Playback** command-line tool to simulate users accessing the application. This Java tool is part of the PlaybackManager project, which must be compiled for the tool to exist.

To play back a recorded session:

- Start the application as you normally would; do *not* use the **-WORecordingPath** flag here). When you start the application you can use adaptors or direct connect.

- Start the **Playback** java tool by entering a command similar to the following:

  ```
  java com.apple.client.playback.Playback -r /tmp/tape1.rec
  ```

  The Playback class must be found in the Java classpath. When the PlayBack Manager project has been compiled, the **Playback** tool bytecode is in the subdirectory **Playback Manager.woa/WebServerResources/Java.**

  Alternatively, you can explictly give the class path on the command line, as in this example:

  ```
  java -classpath
  ".:/MyProjects/PlaybackManager/PlaybackManager.woa/WebServerResources/Ja
  va:'javaconfig DefaultClasspath'" com.apple.client.playback.Playback -r
  /tmp/tape1.rec
  ```

The **Playback** tool plays the recorded session repeatedly until you explicitly stop it (for example, by pressing Control-C in a command shell window). You can run several instances of the tool at the same time to put more load on the server. To manage multiple instances it's better to use the Playback Manager application.

If you want, you can specify other options of the **Playback** tool. The following list describes these options:

-h *hostname*
: Sets the host to send the requests to (the default is **localhost**).

-p *adaptorPath*
: Sends requests using the specified 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 whose adaptor is named **WebObjects.dll.** Your adaptor path is **/Scripts/WebObjects.dll.**

-port *portNumber*
: Sets the port the requests are sent to (the default is 80).

-c *limit*
: Limits the number of times to repeat the session playback (there is no limit by default).

-s *sleepTime*
: Sets the interval between requests in seconds (the default is zero).

-diff *percents*
: Sets the percentage difference between received and recorded response sizes (the default is 5%).

-d
: Turns debugging on.

-r *recordingDir*
: Sets the recording directory.

-help
: Prints a summary of options

Here is an example of a command beginning a playback session using direct connect:

```
java -classpath com.apple.client.playback.Playback -d -h mymachine -r
/tmp/tape1.rec -port 3456 -diff 20
```

Improving Performance

Performance is a major concern of web site 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 "Automatic Scheduling" 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.

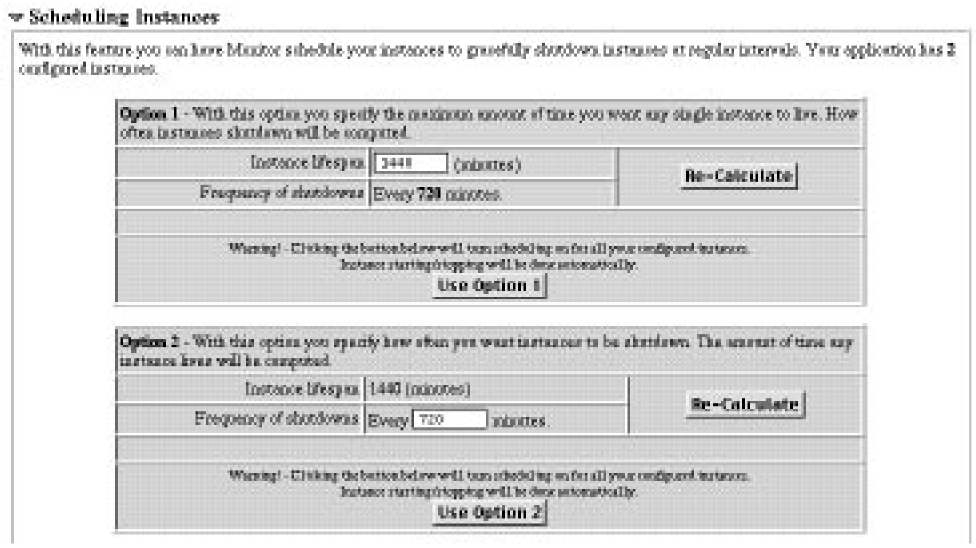
You can also try to reduce the size of the application instance by limiting the amount of state that it stores. Set the session time-out value to ensure that sessions expire after a reasonable length of time. Shut down and restart the application more often to reduce its size.

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.

Automatic Scheduling

You can use Monitor to start and stop instances automatically at regular intervals. Typically, WebObjects applications can run for long periods of time, even months. If your application caches data or has memory leaks, you can schedule it to recycle its instances without interrupting service to your customers.

Use the Scheduling Instances form of the Application Configuration page to configure a pool of instances. This form allows you establish a staggered schedule for stopping and restarting the instances. Here is an example of the Scheduling Instances form:



With this feature you can have Monitor schedule your instances to gracefully shutdown instances at regular intervals. Your application has 2 configured instances.

Option 1 - With this option you specify the maximum amount of time you want any single instance to live. How often instances shutdown will be computed.

Instance lifespan	1440 (minutes)	Re-Calculate
Frequency of shutdowns	Every **720** minutes.	

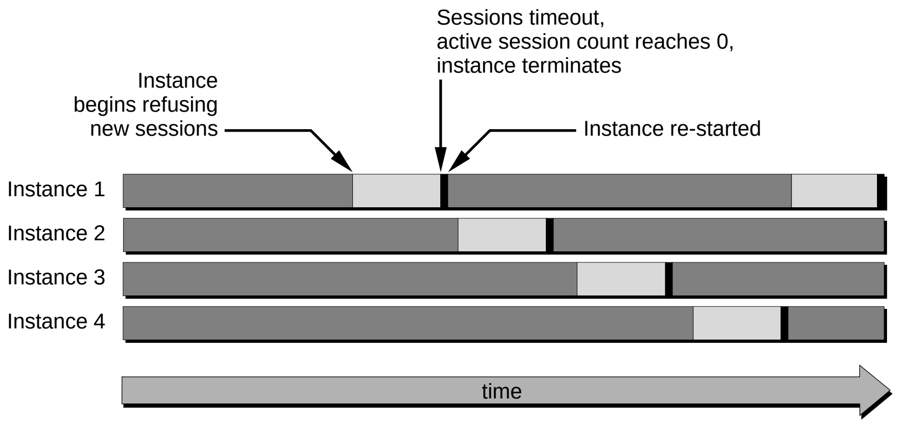
Warning! - Clicking the button below will turn scheduling on for all your configured instances.
Instance starting/stopping will be done automatically.

Option 2 - With this option you specify how often you want instances to be shutdown. The amount of time any instance lives will be computed.

Instance lifespan	1440 (minutes)	Re-Calculate
Frequency of shutdowns	Every 720 minutes.	

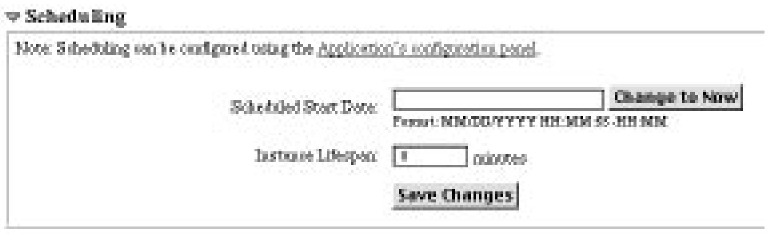
Warning! - Clicking the button below will turn scheduling on for all your configured instances.
Instance starting/stopping will be done automatically.

Either specify the instance lifespan or the frequency of shutdown (both in minutes) and then click the appropriate Use Option button. Each instance runs for a specified period before it begins refusing new sessions, and then it shuts down when the minimum active session threshold is reached. The diagram below displays an example schedule for four instances.



Do not set the frequency of shutdowns too low. If the session time-out for your application is 30 minutes, then the frequency of application shutdowns should not be less than 30 minutes. It should probably be several times higher than that. These settings are configurable because each application may have different needs.

You can also schedule instances individually with the Scheduling option of the Instance Configuration page (to go to this page, click Config next to an instance on the Detail View page):



Note: Scheduling can be configured using the Application's configuration panel.

Scheduled Start Date:
Format: MM/DD/YYYY HH:MM:SS -HH:MM

Instance Lifespan: 8 minutes

Specify the start date (in the recommended format) and the lifespan of the instance in minutes, then click Save Changes

If you have set up scheduling for an application and then add a new instance, the new instance does not have a schedule that is synchronized with the other instances. To insert this new instance into the schedule you need to go to the Application Configuration page and reset the schedule, or you must manually create the schedule in the Instance's Configuration page.

You can programmatically set up an application to shut down in addition to scheduling shutdowns using the Monitor. If you want to use internal scheduling algorithms in your instance, it is not recommended that you also use Monitor's scheduling features. Instead, just use Monitor to recover failures of your instances and to access statistics.

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
```

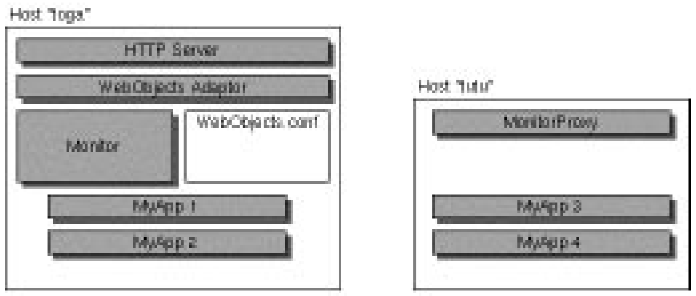
the WebObjects adaptor looks for an instance of MyApp on the host toga. If it finds an instance and the instance is ready to receive requests, the adaptor sends the request to that instance. If both of the instances of MyApp on toga are busy, it accesses an instance on the host tutu.

Use the Monitor application to create multiple new instances of an application for load balancing. See "Creating Application Instances" on page 14 for details.

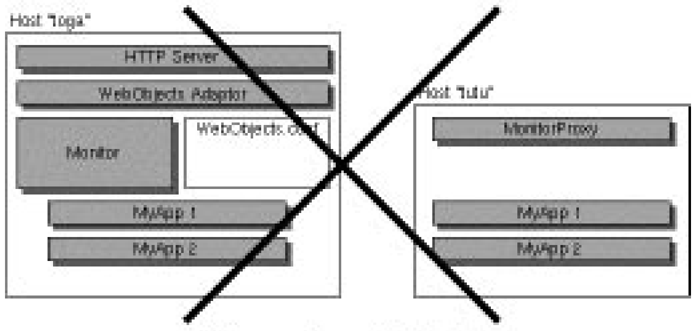
When you create multiple application instances, you are creating the public configuration file *NEXT_ROOT*/**Library/WebObjects/Configuration/WebObjects.conf**. When the adaptor receives an HTTP request for an application, it first (in its initial mode) checks **WebObjects.conf** for an application instance 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.

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.



Right -- instance numbers unique across hosts



Wrong -- not supported by Monitor

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 Instance Detail View page 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.

Increasing the Listen Queue Depth

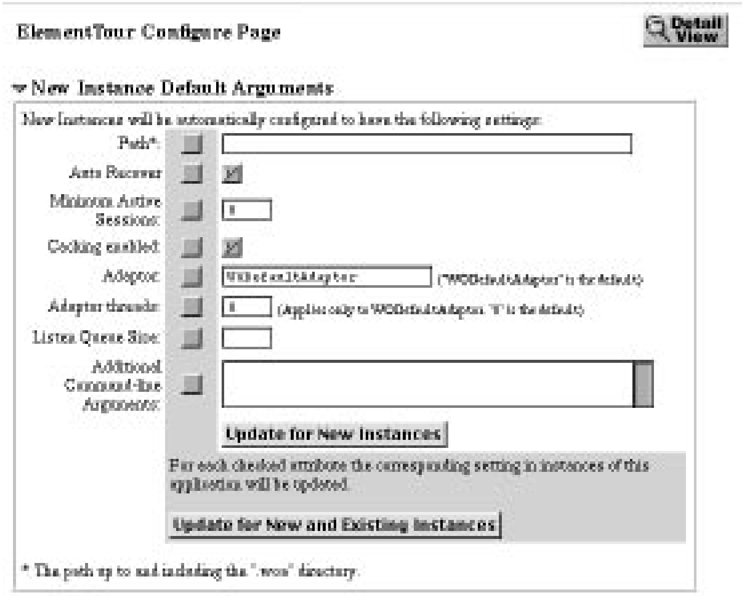
The listen queue depth indicates the number of transactions that can be in the socket buffer (the listen queue) awaiting processing. If the number of transactions in the buffer reach the limit set by the listen queue depth, the socket refuses new requests. The default depth is five.

When an application's request load varies by period (that is, it experiences "spikes"), 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 set the listen queue depth for all instances of an application, do the following in Monitor:

1. Click Applications in the Monitor banner to go to the Applications page.
2. Click the Config button in any row containing a configured application.
3. In the Application Configuration page, click the triangle next to the New Instance Default Arguments option; this discloses the following form:



ElementTour Configure Page

New Instance Default Arguments

New Instances will be automatically configured to have the following settings:

Path*:	
Auto Recover:	✓
Minimum Active Sessions:	0
Caching enabled:	✓
Adaptor:	WODefaultAdaptor ("WODefaultAdaptor" is the default)
Adaptor threads:	8 (Applies only to WODefaultAdaptor. "8" is the default)
Listen Queue Size:	
Additional Command-line Arguments:	

For each checked attribute the corresponding setting in instances of this application will be updated.

* The path up to and including the ".woa" directory.

4. Type the new listen queue depth in the Listen Queue Size field.
5. Click the Update for New and Existing Instances button.
6. Restart any existing instance to have it assume the new listen queue depth.

If you want to change the listen queue depth for specific instances, enter the new depth in the List Queue Size field of the Application Start-Up/Command-line Arguments form in the Instance Configuration page for an instance.

Making Monitor and MonitorProxy Fail-safe

Because Monitor is a critical piece of any deployment, you should take measures to make sure that it does not fail. To help you achieve this aim, WebObjects provides a simple command-line tool, **MonitorDaemon**. This tool restarts Monitor or **MonitorProxy** when they fail. How you use MonitorDaemon depends on the WebObjects deployment platform.

Using MonitorDaemon on Windows NT

When WebObjects is properly installed, the Services Control panel contains two services which use **MonitorDaemon** to keep Monitor and **MonitorProxy** running.: “Apple WebObjects Monitor” and “Apple WebObjects MonitorProxy.” Use these services to keep Monitor and **MonitorProxy** up and running.

To make Monitor and **MonitorProxy** start automatically at boot time, you can configure the services to be started Automatically.

Using MonitorDaemon on Mac OS X Server

On these platforms you can enter the **MonitorDaemon** tool on a shell command line (such as provided by **Terminal.app**), start it from a shell script, or configure it to launch Monitor or **MonitorProxy** automatically at boot time.

For command line usage you pass as arguments the path to the application you want to be launched and then any arguments you want to launch it with. So to start **MonitorDaemon** for Monitor, you might give the following command:

```
MonitorDaemon
/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 and **MonitorProxy** near the end of the boot cycle.

You could add the following script, named **3000_Monitor**, to **/etc/startup** to start **MonitorDaemon** when the system boots and have it keep Monitor running:

```
#!/bin/sh

#
# Start Monitor using MonitorDaemon for WebObjects Deployment
#
. /etc/rc.common

# the following is one line:
/System/Library/WebObjects/Applications/Monitor.woa/MonitorDaemon
/System/Library/WebObjects/Applications/Monitor.woa/Monitor &
```