Deploying WebObjects Applications

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Table of Contents

Introduction

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.

Related Documentation

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.

- Installation instructions for supported HTTP adaptors can be found in **InstallationInstructions.html**, which is located in ***NEXT_ROOT*/Developer/Examples/WebObjects/Source/Adaptors/**. Instructions for building HTTP adaptors from provided source code are located in **BuildingInstructions.html** in the same directory.

WebObjects HTTP Adaptors

A key part of WebObjects administration involves 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 them 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.

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 on 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, CGI adaptors and API-based adaptors. 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. To activate an API-based adaptor, build and install it using the instructions found in **BuildingInstructions.html** and **InstallationInstructions.html** (both are located in ***NEXT_ROOT*/Developer/Examples/WebObjects/Source/Adaptors/**).

CGI Adaptors

The CGI adaptor is an executable file named **WebObjects**—**WebObjects.exe** on Windows NT—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

API-based adaptors are WebObjects adaptors based on APIs specific to a 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 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 as an example project, although 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.

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/Adaptors**. 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.

Adaptor	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

Adaptor	Server	/Library/WebObjects/ Location	Executable
Apache	1.3	Adaptors/Apache	mod_WebObjects.o
WAI	Netscape 3.5 servers	Adaptors/WAI	(must build example project)

You can find installation instructions for supported HTTP adaptors in ***NEXT_ROOT*/Developer/Examples/WebObjects/Source/Adaptors/InstallationInstructions.html**. To build HTTP adaptors from provided source code, refer to ***NEXT_ROOT*/Developer/Examples/WebObjects/Source/Adaptors/BuildingInstructions.html**.

Configuration Files

Web server adaptors obtain configuration information from **wotaskd** via HTTP requests. Web server adaptors default to **wotaskd** on **localhost**, or, when **wotaskd** cannot be found, the adaptor gets its configuration information from ***NEXT_ROOT*/Local/Library/WebObjects/Configuration/WebObjects.xml** (*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. This configuration file isn't present on your system by default; if you need it, you'll need to create this file by hand following the format outlined in "Web Server Adaptor Configuration File Format" (page 11).

In general, you want one 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 configuration file.

If you have multiple HTTP servers running on a single machine, they all share the configuration file. If you want each server to have its own configuration file, you can install one **WebObjects.xml** 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.

The Monitor application also maintains a configuration file for the purposes of crash recovery; in the event that Monitor fails for any reason, when restarted it

reads the file ***NEXT_ROOT*/Local/Library/WebObjects/Configuration/SiteConfig.conf** in order to restore its state.

Automatic Discovery of WebObjects App Servers

This beta release includes support for the automatic discovery of systems running WebObjects by web server adaptors. This should remove the necessity to administer the web servers (beyond the initial adaptor installation).

When the web server adaptor starts up, and at intervals determined by the configuration refresh interval setting, the adaptor sends out a multicast request in an effort to discover which WebObjects app servers are available. Each app server's `wotaskd` process replies with its URL (`http://me.myself.com:1085`). The adaptor constructs a list of these URLs and then polls each in turn to get the full site configuration information.

If the configuration refresh interval is 10 seconds, the discovery broadcast happens every 100 seconds (the discovery broadcast occurs a factor of 10 less frequently).

To enable this automatic discovery mechanism, you need to make changes in both the web server adaptor 's configuration file and on each machine running `wotaskd`.

Web Server Adaptor

The adaptor sends discovery requests out on a particular multicast "channel" (IP address + port). The defaults are:

Default IP Address: `239.128.14.2`

Default port: `1085`

The default multicast address is within the "Adminstratively Scoped Domain." That is, it's within the range of addresses intended for internal use inside organizations.

For Apache, place the following in your `apache.conf` (the final value—10 in this instance—indicates the configuration refresh interval):

```
WebObjectsConfig webobjects://239.128.14.2:1085 10
```

For CGI, either recompile, or set the `WO_CONFIG_URL` environment variable as above.

Note: With Apache, you'll need the `SetEnv` command, which comes with the "env" module. Note that Mac OS X Server doesn't switch this module on by default.

For NSAPI, place something like the following in your `obj.conf`:

Standard:

```
Init fn="WebObjects_init" root="/opt/ns-home/docs"
    config="http://localhost:1085"
```

Multicast:

```
Init fn="WebObjects_init" root="/opt/ns-home/docs"
    config="webobjects://239.128.14.2:1085"
```

For ISAPI, add the following to the registry:

```
\\SOFTWARE\\Apple\\WebObjects\\Configuration\\CONF_URL
    webobjects://239.128.14.2:1085
```

wotaskd

By default `wotaskd` listens for multicast discovery requests on IP address `239.128.14.2`. If you've configured the web server adaptor to send such requests to a different IP address, you must also set the `WOConfigMulticastAddress` user default on machines running `wotaskd`. You must to do this as root/administrator on the given machine (the user running `wotaskd`), or modify the startup script to provide it as a command-line option:

```
defaults write wotaskd WOConfigMulticastAddress 239.128.14.2
```

Web Server Adaptor Configuration File Format

This section details the new web server adaptor configuration file format. Prior to this release, the configuration file was formatted as a property list. For this release, is formatted using XML.

XML Format in Full

The following is a complete definition of the XML-format configuration file, showing all possible attributes and values.

```
<!DOCTYPE WebObjectsAdaptorConfiguration SYSTEM “woadaptor.dtd”>

<adaptor
    retries=NUMBER
    scheduler=[“random”|”roundrobin”|”loadaverage”]
    dormant=NUMBER
    protocol=”http”
    transport=[“socket”|”fsocket”|”winsock”|”nssocket”]
    redir=URL
    xyzzy=STRING
    confinterval=NUMBER
    timeout=NUMBER
    sendTimeout=NUMBER
    recvTimeout=NUMBER
    cnctTimeout=NUMBER
    poolSize=NUMBER
    sendBufSize=NUMBER
    recvBufSize=NUMBER
    urlVersion=[“3”|”3.5”|”4”]
>
</adaptor>
<application
    name=STRING
    retries=NUMBER
    scheduler=[“random”|”roundrobin”|”loadaverage”]
    dormant=NUMBER
    protocol=”http”
    transport=[“socket”|”fsocket”|”winsock”|”nssocket”]
    redir=URL
    timeout=NUMBER
    sendTimeout=NUMBER
    recvTimeout=NUMBER
    cnctTimeout=NUMBER
    poolSize=NUMBER
    urlVersion=[“3”|”3.5”|”4”]
>
<instance
    id=NUMBER
    port=NUMBER
    host=STRING
    dormant=NUMBER
    protocol=”http”
    transport=[“socket”|”fsocket”|”winsock”|”nssocket”]
    redir=URL
    timeout=NUMBER
    sendTimeout=NUMBER
    recvTimeout=NUMBER
    cnctTimeout=NUMBER
    poolSize=NUMBER
    urlVersion=[“3”|”3.5”|”4”]
    refuseNewSessions=[“YES”|”NO”]
>
</instance>
</application>
```

Sections

The configuration file is divided into three sections, as follows:

<adaptor>

In this optional section, specify global default values for applications and instances. Attribute values defined here apply to global adaptor behavior and apply to all applications.

The “error” attribute is used to redirect requests for which no application can be found.

<application>

Required attribute: “name”.

Attribute values defined here specify load balancing behavior and default values for instances of this application.

<instance>

Required attributes: id, port, host

Attribute values defined here specify communication options for this instance and override anything specified at the application.

Attributes

The various attributes used throughout the configuration file are defined as follows:

retries=*NUMBER*

Specifies the number of times to try a request against an application (trying several instances) before returning an error.

scheduler=[“random”|”roundrobin”|”loadaverage”]

Specifies which load balancing algorithm to use to select an application instance

dormant=*NUMBER*

If an instance doesn't respond, do not try to contact it for this many subsequent requests

protocol="http"

The RPC protocol to use to the application. Currently only HTTP is supported.

transport=["socket"|"fsocket"|"winsock"|"nssocket"]

The socket API used to contact an instance. "socket" indicates simple, cross platform, unbuffered sockets. "fsocket" specifies sockets buffered using fopen(), fread(), fwrite() & such (valid on Unix only). "winsock" specifies Win32 socket API (valid on NT only). "nssocket" indicates Netscape's NSAPI socket API (NSAPI only).

redir=*URL*

If an error occurs during request processing, return a redirect (302) HTTP response with *URL* as the location

xyzzy=*STRING*

Return a page reporting some adaptor details when a request for this application arrives

confinterval=*NUMBER*

How often, in seconds, the adaptor should check to see if the configuration has changed

timeout=*NUMBER*

Default for sendTimeout, recvTimeout and cnctTimeout

sendTimeout=*NUMBER*

Timeout, in seconds, before reporting a failed send() to an instance

recvTimeout=*NUMBER*

Timeout, in seconds, before reporting a failed recv() from an instance

cnctTimeout=*NUMBER*

Timeout, in seconds, before reporting a failed connect() to an instance

poolSize=*NUMBER*

Number of persistent connections to maintain with an instance

sendBufSize=*NUMBER*

Size of the TCP/IP socket send buffer (in bytes) that's used for adaptor-to-web-app communication

recvBufSize=*NUMBER*

Size of the TCP/IP socket receive buffer (in bytes) that's used for adaptor-to-web-app communication

urlVersion=["3"|"3.5"|"4"]

The WebObjects version to use for URL parsing and formatting

Sample Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE adaptor SYSTEM "woadaptor.dtd">
<adaptor>
    <application name="HelloWorld"
            retries="5"
            loadbalance="roundrobin"
            dormant="300"
            protocol="http"
            transport="fsocket"
            redir="http://www.apple.com">
        <instance id="1" host="localhost" port="2001"
            refuseNewSessions="NO"
            sendTimeout="3"
            recvTimeout="10">
        </instance>
        <instance id="2" host="localhost" port="2002"
            refuseNewSessions="YES">
        </instance>
    </application>
```

```
    <application name=”Movies”
            urlVersion=”3.5”
            retries=”1”
            loadbalance=”random”
            protocol=”http”
            transport=”socket”>
        <instance id=”3” host=”localhost” port=”1001”
            refuseNewSessions=”NO”>
        </instance>
        <instance id=”4” host=”localhost” port=”1003”
            refuseNewSessions=”NO”>
        </instance>
    </application>

</adaptor>
```

Configuration File DTD

```
<!-- Can actually incorporate status info in here -->

<!ELEMENT adaptor (application)*>
<!ATTLIST adaptor
    xyzzy CDATA #IMPLIED
    confinterval CDATA #IMPLIED
    retries CDATA #IMPLIED
    loadbalance (“random”|”roundrobin”|”loadaverage”) #IMPLIED
    dormant CDATA #IMPLIED
    protocol CDATA “http” #IMPLIED
    transport (“socket”|”fsocket”|”winsock”|”nssocket”) #IMPLIED
    redir CDATA #IMPLIED
    timeout CDATA #IMPLIED
    sendTimeout CDATA #IMPLIED
    recvTimeout CDATA #IMPLIED
    cnctTimeout CDATA #IMPLIED
    sendBufSize CDATA #IMPLIED
    recvBufSize CDATA #IMPLIED
    poolsize CDATA #IMPLIED
    urlVersion (“3”| “3.5” | “4”) #IMPLIED>

<!ELEMENT application (instance)*>
<!ATTLIST application name STRING #REQUIRED
    retries CDATA #IMPLIED
    loadbalance (“random”|”roundrobin”|”loadaverage”) #IMPLIED
    dormant CDATA #IMPLIED
    protocol CDATA “http” #IMPLIED
    transport (“socket”|”fsocket”|”winsock”|”nssocket”) #IMPLIED
```

```
    redir CDATA #IMPLIED
    timeout CDATA #IMPLIED
    sendTimeout CDATA #IMPLIED
    recvTimeout CDATA #IMPLIED
    cnctTimeout CDATA #IMPLIED
    sendBufSize CDATA #IMPLIED
    recvBufSize CDATA #IMPLIED
    poolsize CDATA #IMPLIED
    urlVersion (“3”| “3.5” | “4”) #IMPLIED>

<!ELEMENT instance>
<!ATTLIST instance id CDATA #REQUIRED port CDATA #REQUIRED host CDATA
#REQUIRED
    refuseNewSessions (“YES”|”NO”) #IMPLIED
    count CDATA “-1” #IMPLIED
    dormant CDATA #IMPLIED
    protocol CDATA “http” #IMPLIED
    transport (“socket”|”fsocket”|”winsock”|”nssocket”) #IMPLIED
    redir CDATA #IMPLIED
    timeout CDATA #IMPLIED
    sendTimeout CDATA #IMPLIED
    recvTimeout CDATA #IMPLIED
    cnctTimeout CDATA #IMPLIED
    sendBufSize CDATA #IMPLIED
    recvBufSize CDATA #IMPLIED
    poolsize CDATA #IMPLIED
    urlVersion (“3”| “3.5” | “4”) #IMPLIED>
```

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 the path to the directory in which the application is to be installed, such as `$(LOCAL_LIBRARY_DIR)/WebObjects/Applications`.

   If you're installing a framework, type
   `$(LOCAL_LIBRARY_DIR)/Library/Frameworks`

Note: You'll need write permission for the directory into which the application is to be installed in order for the build to succeed.

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

   INSTALLDIR_WEBSERVER

Note: You'll need write permission for the WebServer's doc root in order for such a "split install" to succeed.

3. In the Project Build panel, click the check-mark 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**, and that you installed your application in *`LOCAL_LIBRARY_DIR`*`/WebObjects/Applications`, the following directories will be created:

```
LOCAL_LIBRARY_DIR/WebObjects/Applications/MyApp.woa
    MyApp[.exe]
    Resources/
    WebServerResources/
```

```
DOC_ROOT/WebObjects/MyApp.woa
    WebServerResources/
```

When the client tries to contact an application, the adaptor first looks for a configuration file that names the application, and then for an executable in ***DOC_ROOT*/WebObjects** and ***NEXT_ROOT*/Library/WebObjects/Applications**. Thus, you can install the entire directory under ***DOC_ROOT*/WebObjects**. However, doing so presents a security problem if you have scripted components, since any client can access any file under the document root. This means that if you install scripted components under the document root, you're exposing source code to outside users.

Instead, it is recommended that you do a "split install", installing 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 ***DOC_ROOT*/WebObjects** directory rather than in a subdirectory. If you install in a subdirectory, your application will still run but won't find image files unless you explicitly provide the application's base URL (WOApplicationBaseURL). For more information, see "Starting Up Applications From the Command Line" (page 37).

Deploying With Monitor

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 are also installed in your web server's document root under ***DOC_ROOT*/WebObjects/Monitor.woa**. Verify that both these paths exist if you have problems starting 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 the server starts up. To do this on Windows NT, open the Control Panel and click the Services icon. Look for a service named "Apple WebObjects Monitor." You can configure this service to start up automatically.

Starting Up Monitor

To start up Monitor, either double-click the Monitor executable (***NEXT_ROOT*/Library/WebObjects/Applications/Monitor.woa/Monitor**[**.exe**]) from the Windows NT Explorer or the Mac OS X Server Workspace Manager, or execute it from within a terminal window (use the Bourne Shell program on Windows

NT). You can run the Monitor application on any machine that is running **wotaskd**.

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

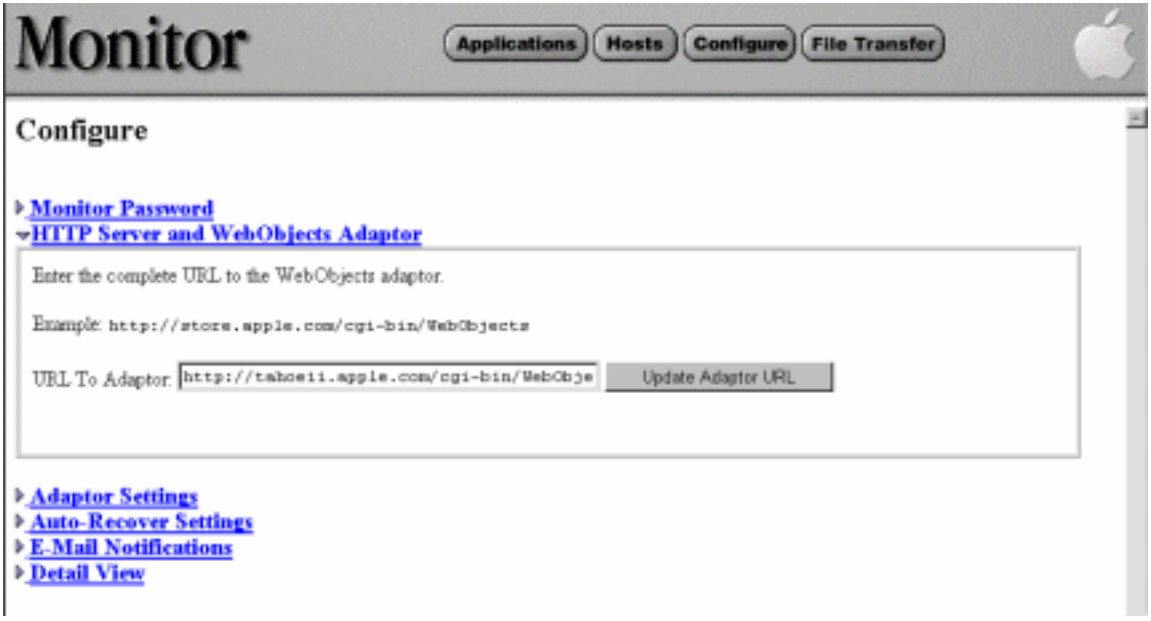
Note that because Monitor is itself a WebObjects application, once Monitor is running you can access it from any client browser. Also note that Monitor can be configured to require a password before client access is granted; if Monitor has been so configured, you'll need to enter the proper password (and click the Login button) before you'll see the Applications Page.

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:

2. Click the "HTTP Server and WebObjects Adaptor" item. Doing so causes the display of the following page:



3. Enter the adaptor's URL in the URL To Adaptor field.

   This URL should identify the WebObjects adaptor running on the web server from which clients will access applications.

4. Click Update Adaptor URL.

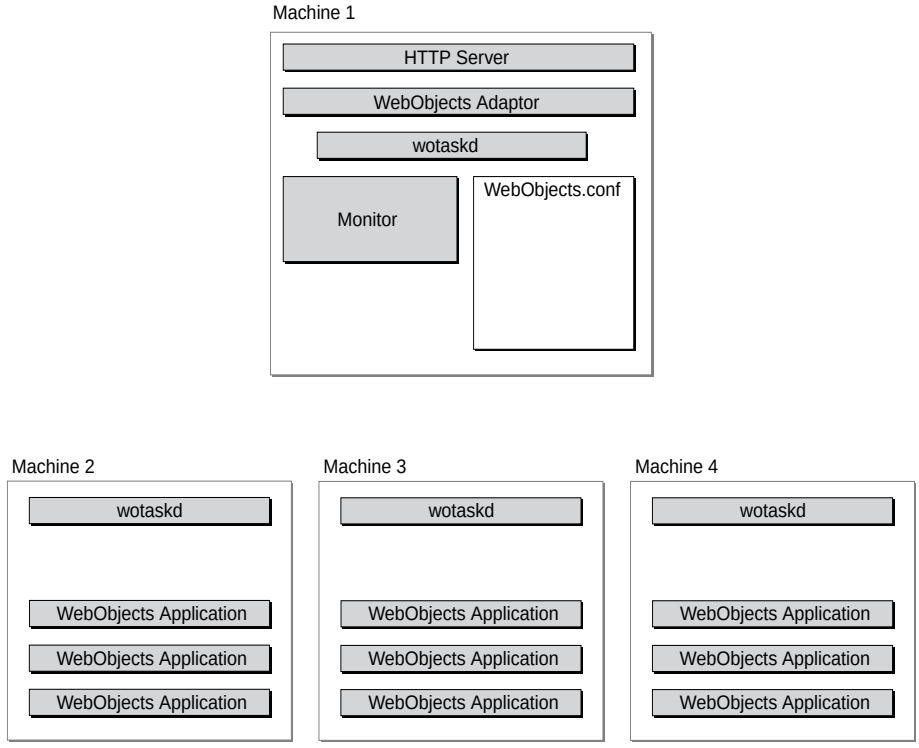
Setting the adaptor URL is the minimal setup task required for administering applications on the local machine. You might want to fine-tune your site's configuration and take advantage of other features such as e-mail notifications.

Deploying on Multiple Hosts

Creating a deployment environment sometimes involves more than one HTTP server and many WebObjects application instances running on each server. To make several hosts available to Monitor, a daemon (or, on NT, a service) called **wotaskd** must be running on each host. With **wotaskd** running the Monitor application can remotely administer a host machine. When WebObjects is installed on a particular host, **wotaskd** is automatically configured to run whenever that host machine is restarted.

Although there is no technical reason why you cannot have multiple copies of Monitor running, because Monitor maintains some information locally it is possible for multiple Monitors managing a common set of hosts to get out of sync. Thus, there ought to be only one copy of Monitor running at a time managing a given set of hosts.

The following diagram depicts one possible WebObjects deployment scenario:

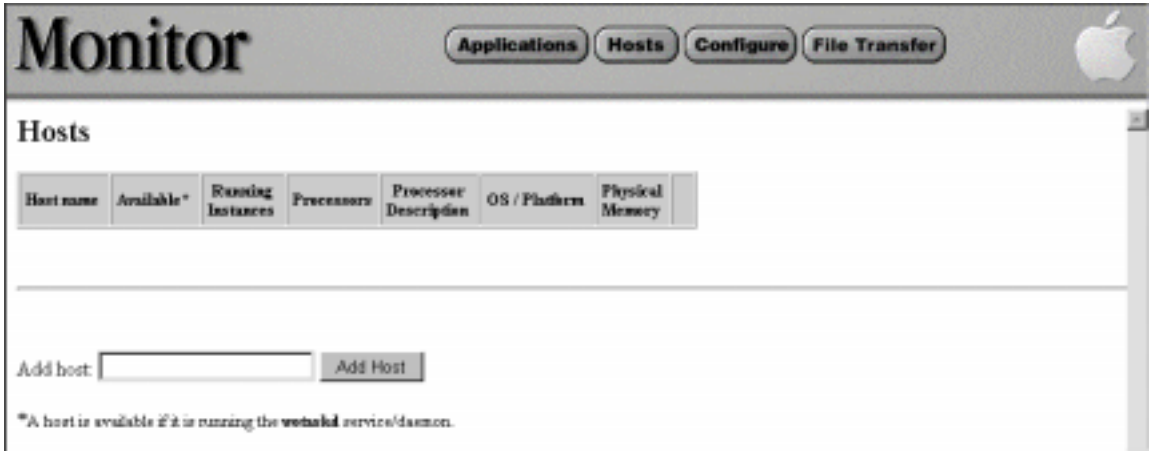


Machine 1 acts as the web server and load balancer between all the application servers running on Machines 2, 3, and 4. Monitor should be running on the same host as the web server and the WebObjects adaptor, and is responsible for maintaining the **WebObjects.conf** file.

Adding a Host to Monitor

Before you can configure WebObjects applications on a remote machine, you should let the Monitor application know about the remote machine. To do this, complete the following steps:

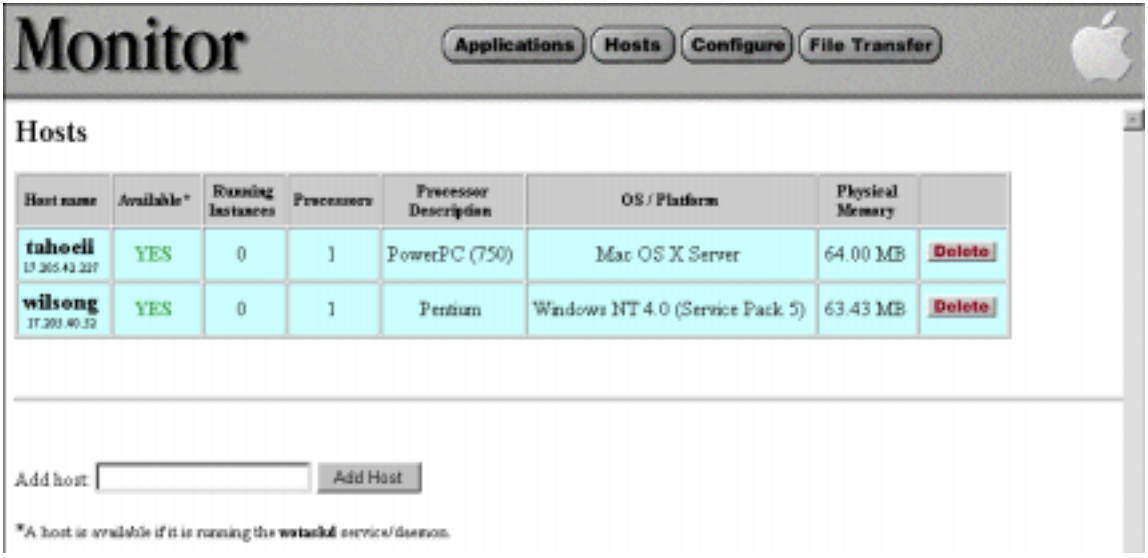
1. Click the Hosts button in the Monitor banner. The following page is displayed:



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, and must be running **wotaskd**).

3. Click Add Host.

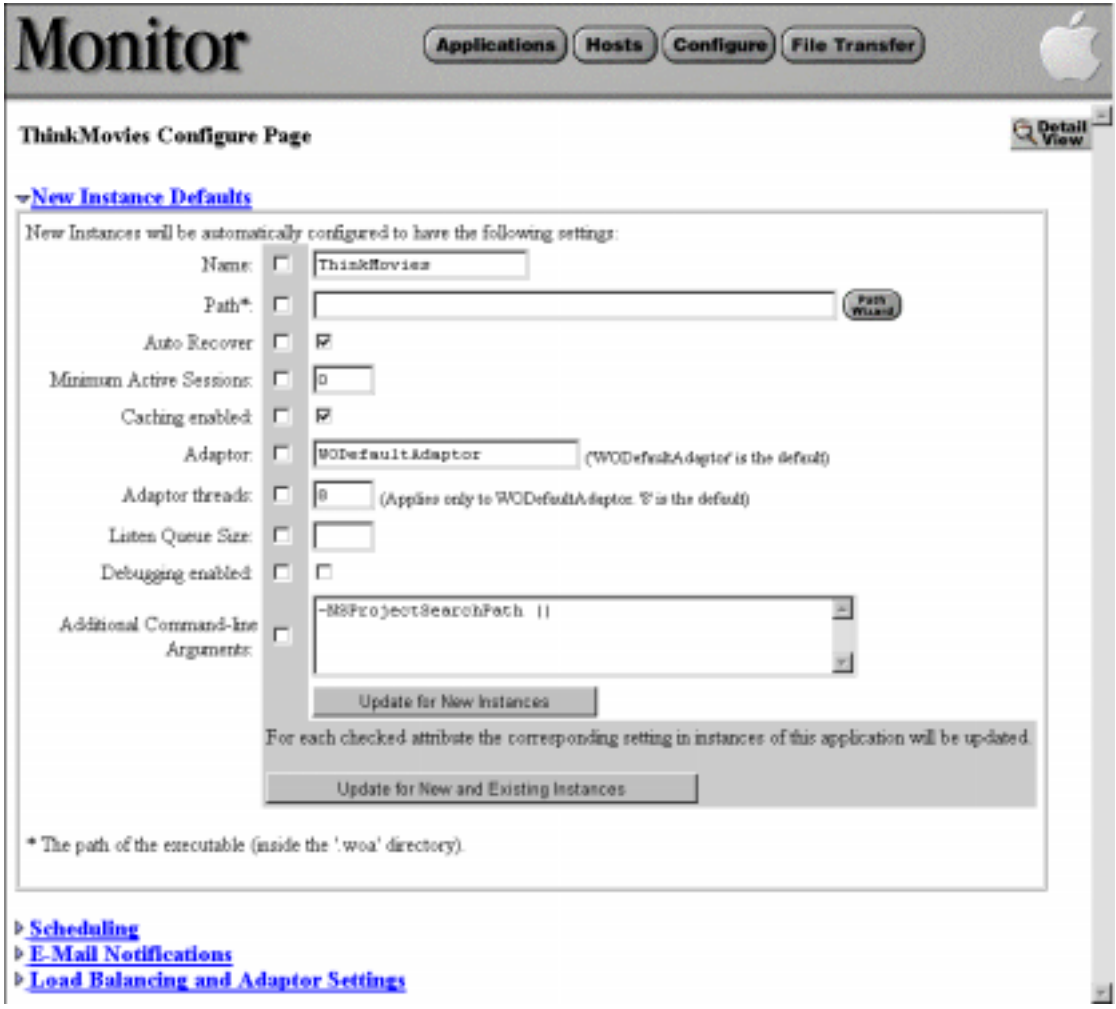
After adding two hosts, Monitor will look something like this:



Adding and Configuring an Application

To add a new application to Monitor, click the Applications button in the banner. This brings you to the Applications page, which lists all currently-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, which is the wrapper name minus the **.woa**. For the ThinkMovies example application, the entered string would be "ThinkMovies".

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



Enter the full path to the WebObjects application executable in the Path field, or use the Path Wizard to select the application's executable file (the Path Wizard allows you to browse the filesystem of any configured host computer and select the application's executable). For ThinkMovies you might enter a string similar to the following example:

```
/WebObjects/ThinkMovies/ThinkMovies.woa/ThinkMovies.exe
```

Be sure that the path specifies the built WebObjects application's executable, including the **.exe** extension if on Windows NT. You cannot start an instance of an application when the wrong path is specified; Monitor will display "Launch

error - path invalid" if you attempt to start such an instance. Click the Update for New Instances button on the bottom of the form to save your changes.

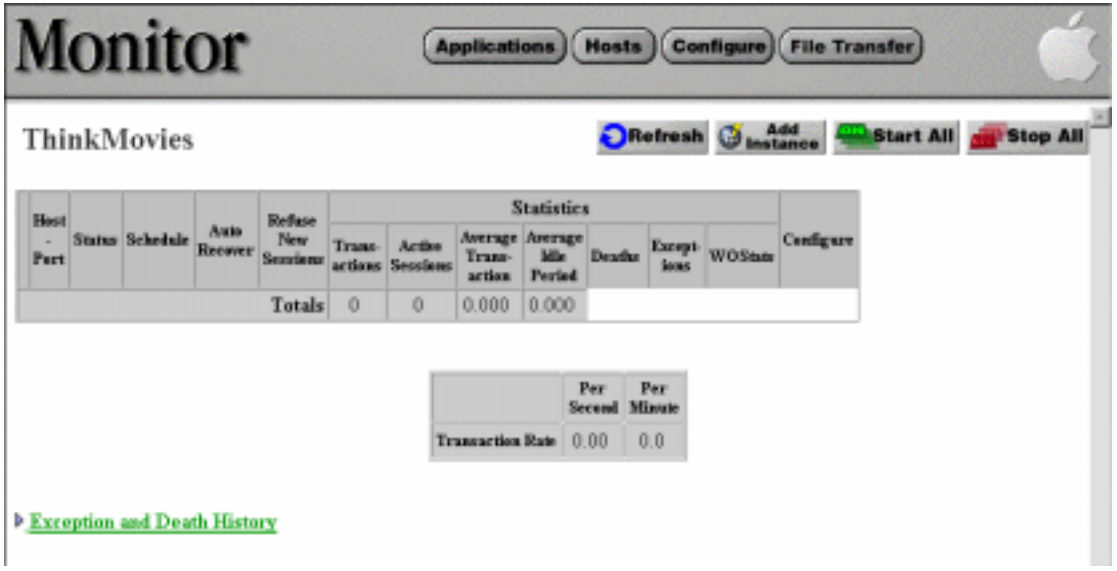
The other fields on this form accept arguments to use when the application instance is run. For descriptions of these fields and as well as the checkboxes and the Update for New and Existing Instances button, see "Setting Command-Line Arguments in Monitor" (page 32) and "Monitor Option Summary" (page 40).

Creating Application Instances

Each application instance you create initially adopts the defaults provided in the Application Configuration page for the application; you can later customize individual instances. 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:



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 along with the number of instances to add.

3. Select the host on which the application resides from the pop-up menu (if the application doesn't reside on the selected host, Monitor will display a "Launch error - invalid path" message when you attempt to start the instance).

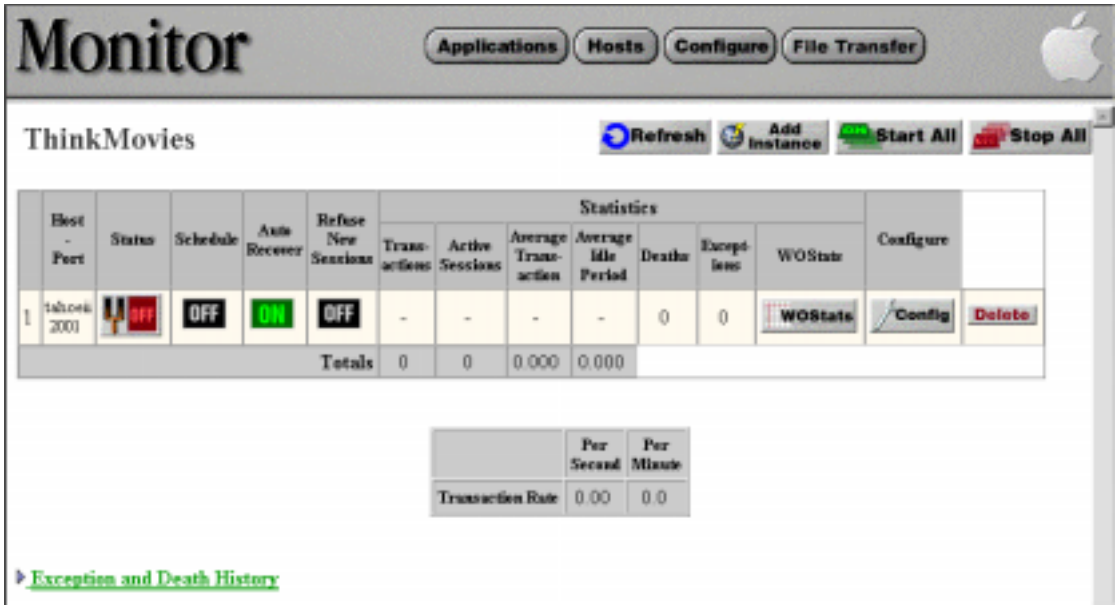
   Unless you previously configured hosts in Monitor, there should only be one item in the pop-up menu.

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.

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. Or, from the Application Configuration page for an application, you can get to the Application Detail View page by clicking the Detail View button in the upper right corner of the page. In either case, a page similar to the following should then appear:



The button that looks like a power switch in the Status column 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” (page 47).

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 (note that it isn’t displayed in your 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 in the web server's error log to find out why.
- Your instance is still starting up and Monitor has not yet received notification of a successful start. Wait a few seconds and refresh the display.
- Monitor couldn't start your instance because the path was wrong or the executable did not exist. In this case, an error message will be displayed above the instances table when the display is refreshed.

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 by first locating a running **wotaskd** daemon or service on the appropriate host. If it finds a **wotaskd**, it passes the application arguments to it. If it cannot contact a **wotaskd**, it does not start the application.

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.

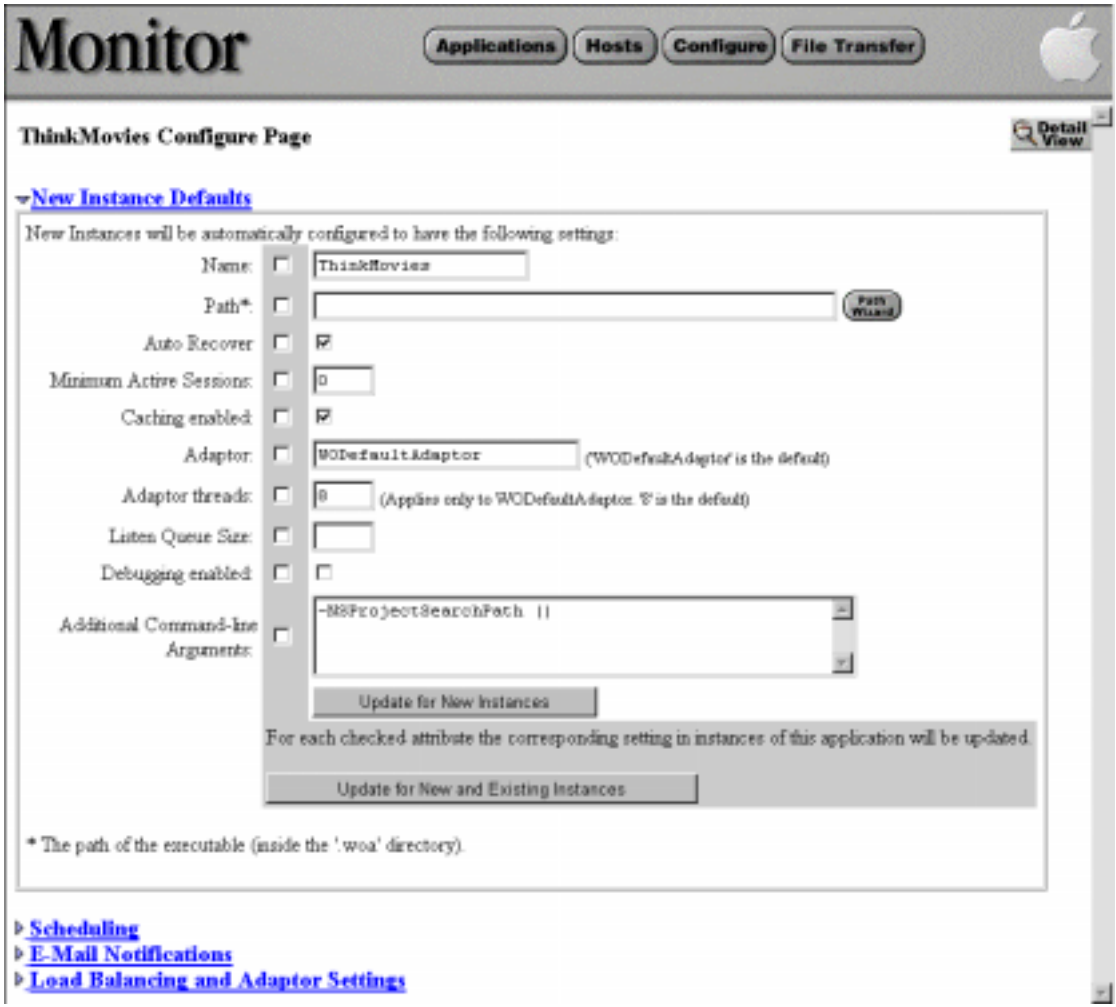
Setting Command-Line Arguments in Monitor

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" (page 37). You can change an applications arguments, even for all instances that are currently configured and running, by doing one of the following:

- **New application**: Display the Applications page by clicking the Applications button in the Monitor banner, enter the name of the application in the Add Application field, and then click the Add Application button.
- **Existing application**: Display the Application Configuration page for the application by clicking the Config button next to the application in the Application's page (which you can get to by clicking the Applications button

in the Monitor banner). Click the arrow next to the New Instance Default Arguments option of the Application Configuration page.

The following form is displayed:



3. Specify the command-line options you want your application's instances to have. The most common options, which can be changed by simply clicking checkboxes or entering values in fields, are:

Field	Option
Name	The name of the application, which is the WebObjects wrapper name minus the ".woa" extension.
Path	The full path to the WebObjects application's executable (including the ".exe" extension on Windows NT).
Auto Recover	Specifies whether Monitor should try to restart the instance if the instance fails.
Minimum Active Sessions	Specifies the minimum number of active sessions allowed.
Caching enabled	Command-line option -WOCachingEnabled. Requests that the application cache component definitions (templates) instead of reparsing HTML and declaration files upon each new HTTP request.
Adaptor	Command-line option -WOAdaptor. The WOAdaptor class name.
Adaptor threads	Command-line option -WOWorkerThreadCount. The maximum number of worker threads for a multithreaded application. Setting this count to 0 results in single-threaded request dispatch.
Listen Queue Size	Command-line option -WOListenQueueSize. The depth of the listen queue If the application is expected to experience "spikes" in its processing load, consider increasing the listen queue depth (althou increasing this setting does not necessarily improve performance or allow the application to server more requests at sustained high loads).
Debugging Enabled	Command-line option -WODebuggingEnabled. Controls whether the application prints debugging messages to standard error duing startup.

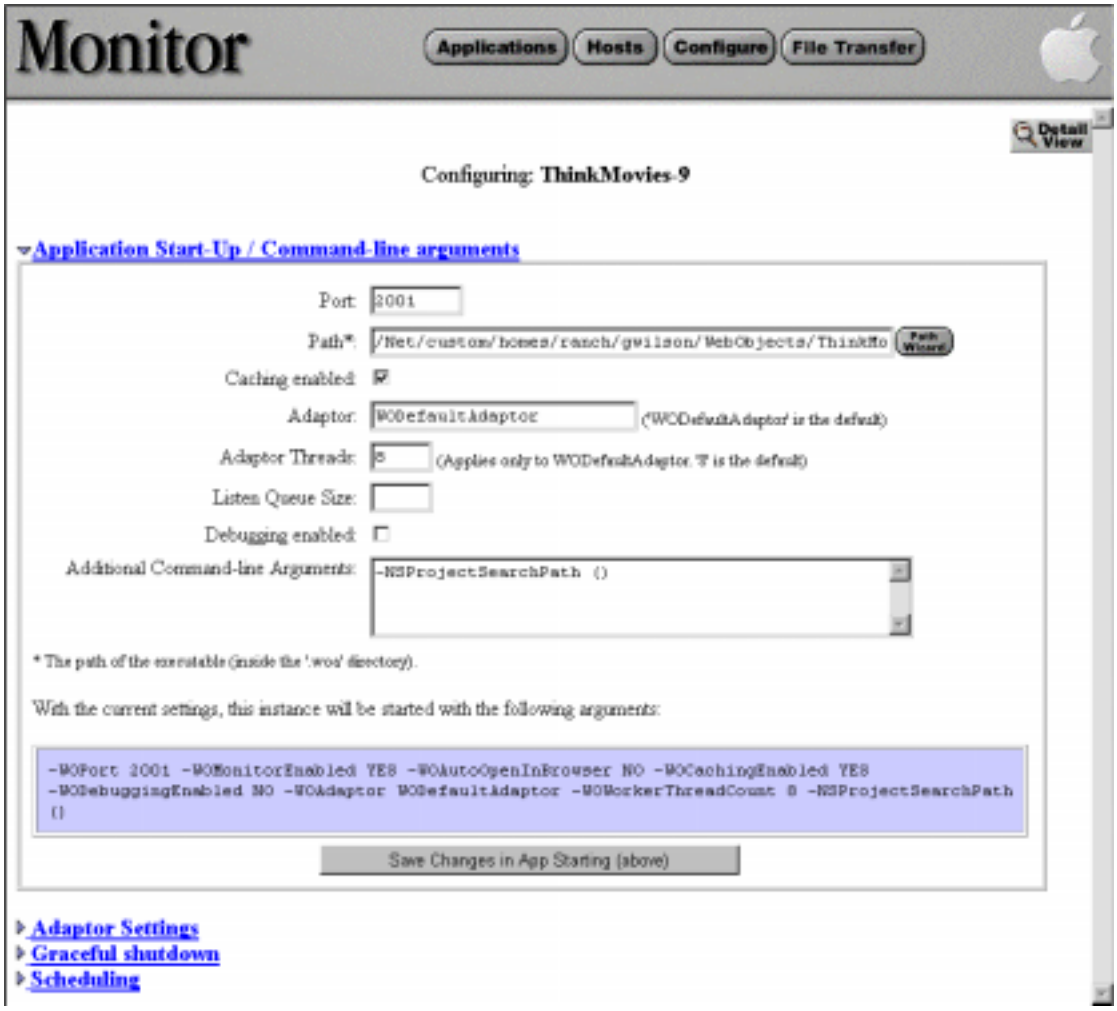
For command-line arguments not included in the above table, enter them as "*-key value*" pairs, separated by spaces, in the Additional Command-line Arguments field. See "Starting Up Applications From the Command Line" (page 37) for a complete list of command-line arguments.

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

will have to be restarted for new options to take effect. To make the changes effective for new instances only, click the Update for New Instances button instead.

Setting Command-Line Arguments for a Specific Instance

The above procedure affects all new or existing instances of a WebObjects application. Monitor also allows you to 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:



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.

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*/Local/Library/WebObjects/Applications** (recommended) or under ***DOC_ROOT*/WebObjects**. For scripted applications, go to the application's **.woa** directory and execute **WODefaultApp**, which is located in ***NEXT_ROOT*/Library/WebObjects/Executables**.

The following table describes each command-line option:

Option	Description
-WODebuggingEnabled YES|NO	Sets whether the application prints messages to standard error during startup. By default, this option is enabled. WOApplication, WOComponent, and WOSession define a **debugWithFormat:** method (**debugString** in Java). This method is similar to **logWithFormat:** except that it only prints messages if the **WODebuggingEnabled** option is on.
-WOAutoOpenInBrowser YES|NO	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 YES|NO	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.

Option	Description
-WOCachingEnabled YES|NO	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 WOMultiThreadedAdaptor.
-WOPort *portNumber*	The socket port used to connect to an application instance. 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 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" (page 65).
-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 ***DOC_ROOT*/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.
-WORecordingPath *path*	Enables session recording and sets the location where the recording file is to be stored. Automatically creates a directory, if one doesn't exist. An extensions of ".rec" is appended to the specified path. See "Recording a Session" (page 54) for more information.

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 ***DOC_ROOT*** 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 could create a shell script that starts WebObjects applications when the server machine is booted. You could also create another shell script that you can run at the command line to start applications.

Monitor Option Summary

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. Click the Configure button in the Monitor banner to display the Global Configuration page.

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.

Adaptor Settings

This section is where you specify global default values for applications and instances (many of these settions can be overridden for individual applications or instances). Attribute values defined here apply to global adaptor behavior and apply to all applications.

Load balancing scheme: Specifies which load balancing algorithm to use to select an application instance.

Transport: The socket API used to contact an instance. “socket” indicates simple, cross platform, unbuffered sockets. “fsocket” specifies sockets buffered using fopen(), fread(), fwrite() & such (valid on Unix only). “winsock” specifies Win32 socket API (valid on NT only). “nssocket” indicates Netscape's NSAPI socket API (NSAPI only).

Adaptor stats string: Return a page reporting some adaptor details when a request for this application arrives.

Configuration read interval: How often, in seconds, the adaptor should check to see if the configuration has changed.

Retries: Specifies the number of times to try a request against an application (trying several instances) before returning an error.

Redirection URL: If an error occurs during request processing, return a redirect (302) HTTP response with the specified URL as the location.

Timeout: Default for Send Timeout, Receive Timeout, and Connect Timeout.

Send timeout: Timeout, in seconds, before reporting a failed send() to an instance.

Receive timeout: Timeout, in seconds, before reporting a failed recv() from an instance.

Connect timeout: Timeout, in seconds, before reporting a failed connect() to an instance.

Connection pool size: Number of persistent connections to maintain with an instance.

Click Change Adaptor Settings to cause changes in the above settings to take effect.

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 **wotaskd** decides 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 enabled, when **wotaskd** is started (perhaps when a machine is booted) it will 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 **wotaskd**, and then have **wotaskd** start all the appropriate applications.

When this setting is on, **wotaskd** perform this check on regular intervals. If **wotaskd** fails or you change an instance's Auto-Recover setting, **wotaskd** 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.

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 the Host Configuration screen, 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 **wotaskd**. 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 24). 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 the Application Configuration page, click the Applications button on the Monitor banner, then click the Config button next to any application.

New Instance Defaults

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" (page 37) 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.

"Setting Command-Line Arguments in Monitor" (page 32) discusses how to set launch options using Monitor.

Scheduling

This section enables the administrator to configure an application's pool of instances (you must have more than one instance running) 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" (page 59) 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 40) 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 email feature, delete all addresses from the text field and click Update.

Load Balancing and Adaptor Settings

This section is where you specify default values for individual applications (many of these settions can be overridden for individual instances). These settings override the corresponding settings made for a given host.

Load balancing scheme: Specifies which load balancing algorithm to use to select an application instance.

Failover mode: Allows you to specify what should happen when an error occurs: report the error to the client, retry the same instance, or try another instance (which one is determined by the load balancing scheme).

Transport: The socket API used to contact an instance. “socket” indicates simple, cross platform, unbuffered sockets. “fsocket” specifies sockets buffered using fopen(), fread(), fwrite() & such (valid on Unix only). “winsock” specifies Win32 socket API (valid on NT only). “nssocket” indicates Netscape's NSAPI socket API (NSAPI only).

Timeout: Default for Send Timeout, Receive Timeout, and Connect Timeout.

Send timeout: Timeout, in seconds, before reporting a failed send() to an instance.

Receive timeout: Timeout, in seconds, before reporting a failed recv() from an instance.

Connect timeout: Timeout, in seconds, before reporting a failed connect() to an instance.

Connection pool size: Number of persistent connections to maintain with an instance.

Click Change Adaptor Settings to cause changes in any of these settings to take affect.

Instance Configuration Options

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

Configuration page, go to the Detail View for the application and 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” (page 32) 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.

Adaptor Settings

This section is where you specify default values for individual instances of an applications. These settings override the corresponding settings made for the application and for the host on which the application instance is running.

Transport: The socket API used to contact an instance. “socket” indicates simple, cross platform, unbuffered sockets. “fsocket” specifies sockets buffered using fopen(), fread(), fwrite() & such (valid on Unix only). “winsock” specifies Win32 socket API (valid on NT only). “nssocket” indicates Netscape's NSAPI socket API (NSAPI only).

Redirection URL: If an error occurs during request processing, return a redirect (302) HTTP response with the specified URL as the location.

Timeout: Default for Send Timeout, Receive Timeout, and Connect Timeout.

Send timeout: Timeout, in seconds, before reporting a failed send() to an instance.

Receive timeout: Timeout, in seconds, before reporting a failed recv() from an instance.

Connect timeout: Timeout, in seconds, before reporting a failed connect() to an instance.

Connection pool size: Number of persistent connections to maintain with an instance.

Click Change Adaptor Settings to cause changes in any of these settings to take affect.

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" (page 59) for the scheduling procedure.

Monitor computes a series of shutdown dates from the desired instance lifespan or from the desired instance downtime. 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:

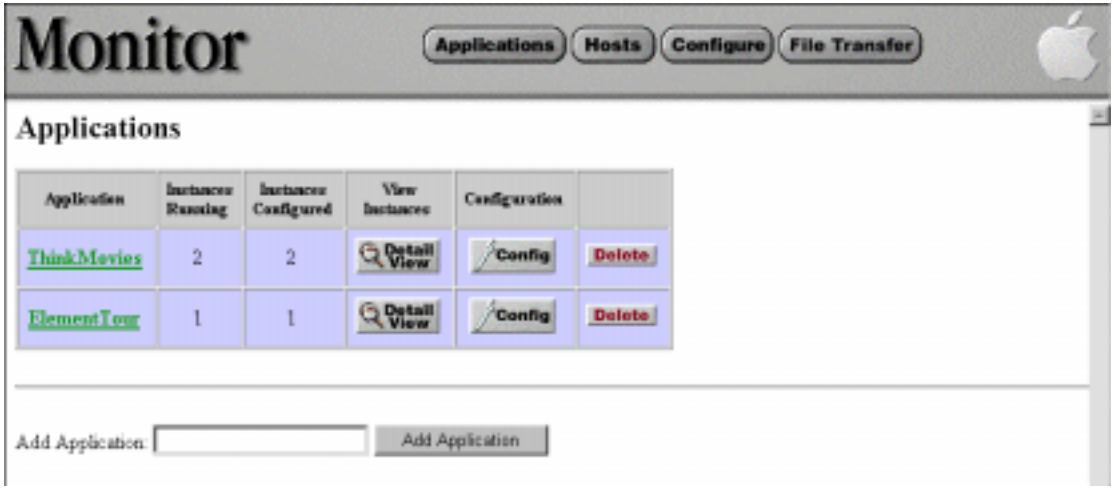
- Monitoring Application Activity
- Performance Testing
- Improving Performance
- Automatic Scheduling
- Load Balancing
- Increasing the Listen Queue Depth
- Making Monitor and wotaskd Fail-safe

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.

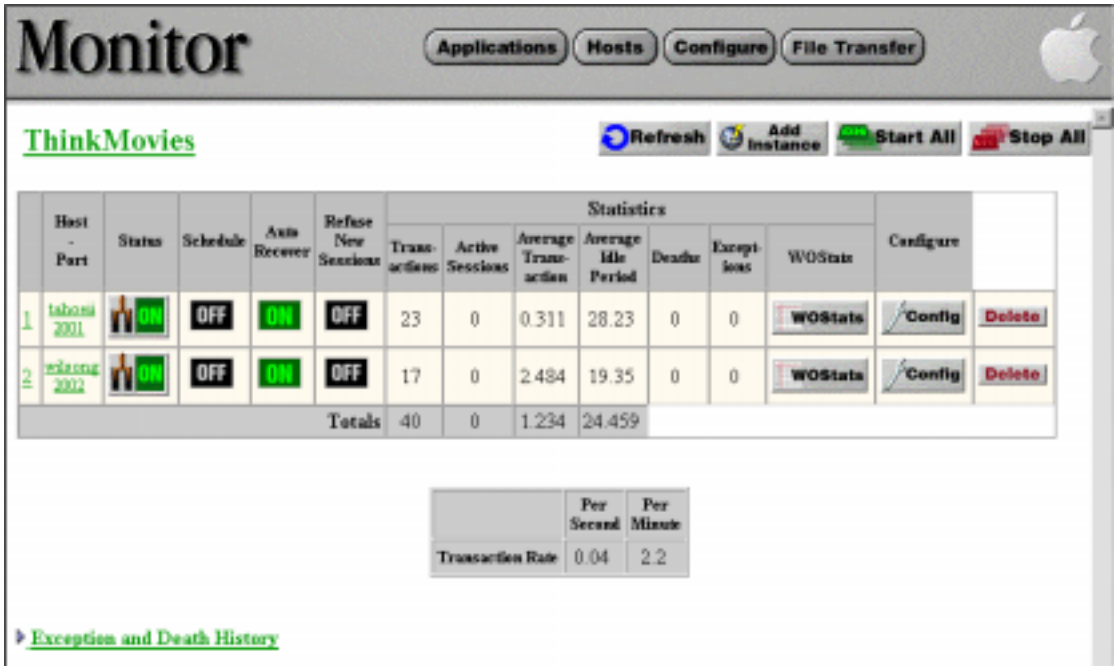
Obtaining Information From Monitor

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:



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:



At the top of the page is the title of the application. When one or more instances of an application are running, 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.

Column	Description
Status	Indicates whether the instance is running (ON) or is stopped (OFF). Clicking this control starts and stops the instance.
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" (page 59)" 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" (page 32).
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.
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 Transaction	The average length, in seconds, of this instance's transactions .
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 view detailed statistics about this instance. See "Accessing the Application Statistics Page" (page 51) for more information.
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.

In addition, the "Transaction Rate" 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.

The following excerpt includes an error message indicating that ThinkMovies wasn't running when a request for it came in:

```
Info: <CGI> new request: /cgi-bin/WebObjects/ThinkMovies
Info: V4 URL: /cgi-bin/WebObjects/ThinkMovies
Error: Request handling error: The requested application was not
found on this server
```

After ThinkMovies was started, the same request produces the following log file entries:

```
Info: <CGI> new request: /cgi-bin/WebObjects/ThinkMovies
Info: V4 URL: /cgi-bin/WebObjects/ThinkMovies
Info: Selecting new app instance
Debug: Composed URL to '/cgi-bin/WebObjects/ThinkMovies.woa/1'
Info: New request is GET /cgi-bin/WebObjects/ThinkMovies.woa/1
HTTP/1.0

Info: Trying to contact ThinkMovies:1 on (2001)
Info: attempting to connect to myhost.apple.com on port 2001
Info: Created new transient connection to myhost.apple.com:2001
Info: ThinkMovies:1 on (2001) connected [pooled: No]
Info: Request GET /cgi-bin/WebObjects/ThinkMovies.woa/1 HTTP/1.0
sent, awaiting response [1 pending]
Info: New response: HTTP/1.0 200 Apple WebObjects
Info: received ->200 Apple
```

Creating the Log File

To cause the log file to be generated, simply create the logWebObjects file in your server's temp directory. The following procedure shows you how to do this:

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.

Use 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 that can access your application.

- 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:

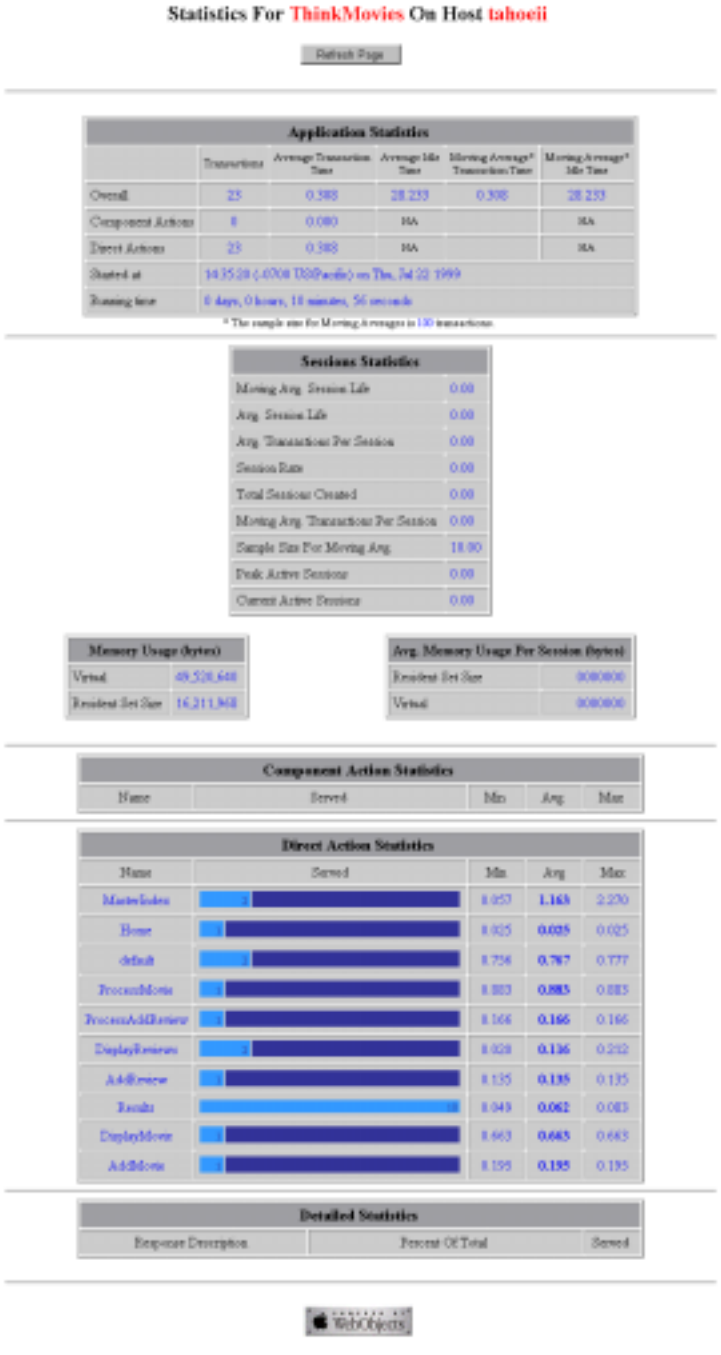
  ```
  http://myhost/cgi-bin/WebObjects/MyWebApp.woa/wa/WOStats
  ```

  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:



Statistics For ThinkMovies On Host tahoeii

Refresh Page

Application Statistics					
	Transactions	Average Transaction Time	Average Idle Time	Moving Average* Transaction Time	Moving Average* Idle Time
Overall	25	0.388	28.253	0.308	28.253
Component Actions	0	0.000	NA		NA
Direct Actions	25	0.388	NA		NA
Started at	14:35:28 (-0700 DST/Pacific) on Thu, Jul 22 1999				
Running time	0 days, 0 hours, 18 minutes, 56 seconds				

* The sample size for Moving Averages is 100 transactions

Sessions Statistics	
Moving Avg. Session Life	0.00
Avg. Session Life	0.00
Avg. Transactions Per Session	0.00
Session Rate	0.00
Total Sessions Created	0.00
Moving Avg. Transactions Per Session	0.00
Sample Size For Moving Avg.	18.00
Peak Active Sessions	0.00
Current Active Sessions	0.00

Memory Usage (bytes)	
Virtual	49,520,640
Resident Set Size	16,211,968

Avg. Memory Usage Per Session (bytes)	
Resident Set Size	8080800
Virtual	8080800

Component Action Statistics				
Name	Served	Min	Avg	Max

Direct Action Statistics				
Name	Served	Min	Avg	Max
MasterIndex	3	0.057	**1.163**	2.270
Home	1	0.025	**0.025**	0.025
default	3	0.756	**0.767**	0.777
ProcessMovie	1	0.883	**0.883**	0.883
ProcessAddReview	1	0.166	**0.166**	0.166
DisplayReviews	3	0.020	**0.136**	0.252
AddReview	1	0.135	**0.135**	0.135
Results	10	0.049	**0.082**	0.083
DisplayMovie	1	0.683	**0.683**	0.683
AddMovie	1	0.195	**0.195**	0.195

Detailed Statistics		
Response Description	Percent Of Total	Served

WebObjects

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**)

These 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 tools expect the application to return the same page and content as 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:

1. Add the following directory to your CLASSPATH environment variable:

   ```
   NEXT_ROOT/Library/WebObjects/Applications/PlaybackManager.woa/WebServerResources/Java
   ```

2. In a separate shell, 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.

3. 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
“.:$NEXT_ROOT/Library/WebObjects/Applications/PlaybackManager.w
oa/WebServerResources/Java:‘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
```

Additional information on the Playback Manager can be found in ***NEXT_ROOT*/Library/WebObjects/Applications/PlaybackManager.woa/Resources/ReadMe.html**.

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. Click "Applications" in the Monitor banner.

b. Click the Config button in the row corresponding to the application for which you want to enable component-definition caching.

c. Click New Instance Defaults.

d. Ensure that Caching Enabled is checked.

e. Click Update for New Instances, or ensure that the left-hand checkbox is also checked and click Update for New and Existing Instances.

- 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" (page 59)" 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" (page 62)" 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" (page 65)" 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.

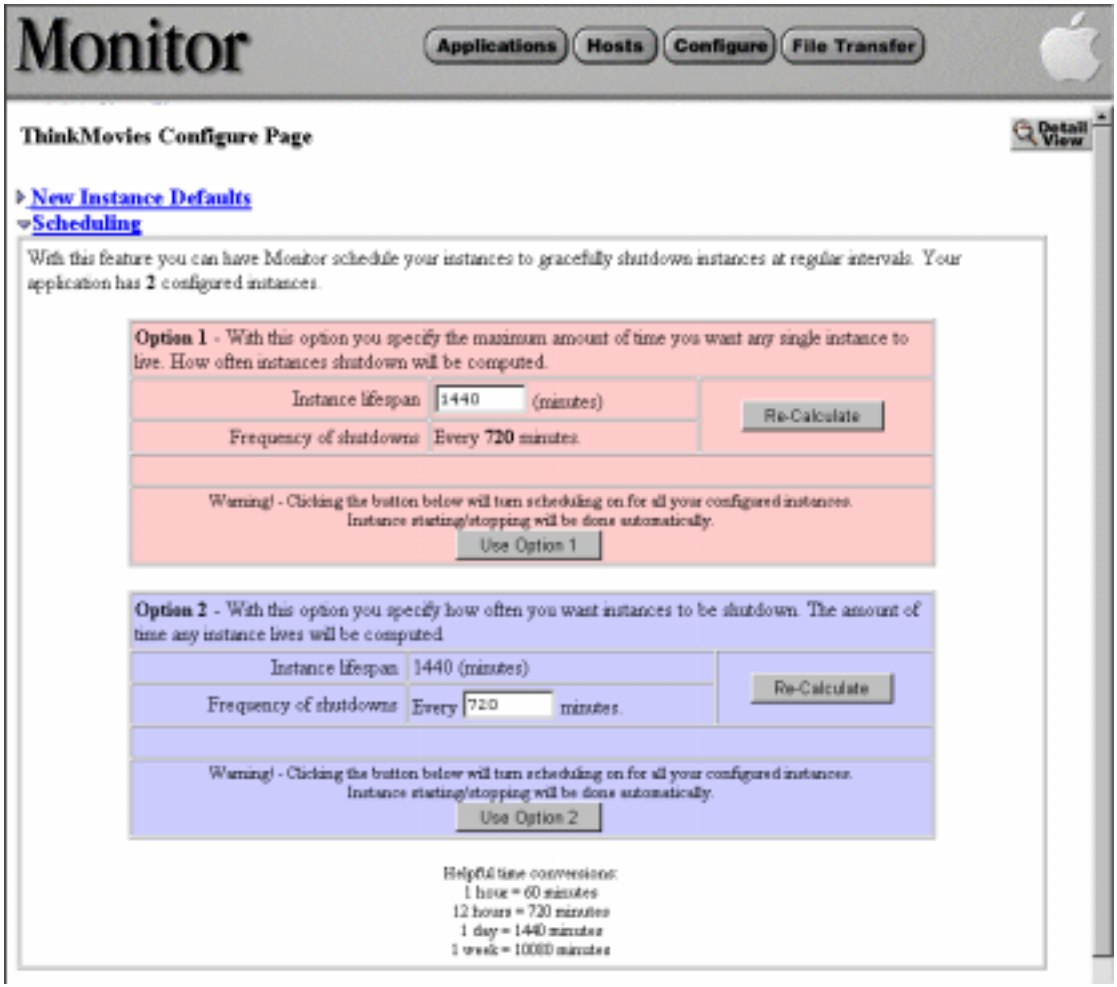
- Try to reduce the size of the application instance by limiting the amount of state that it stores. Set the session time-out value to ensure that sessions expire after a reasonable length of time. Shut down and restart the application more often to reduce its size.

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.

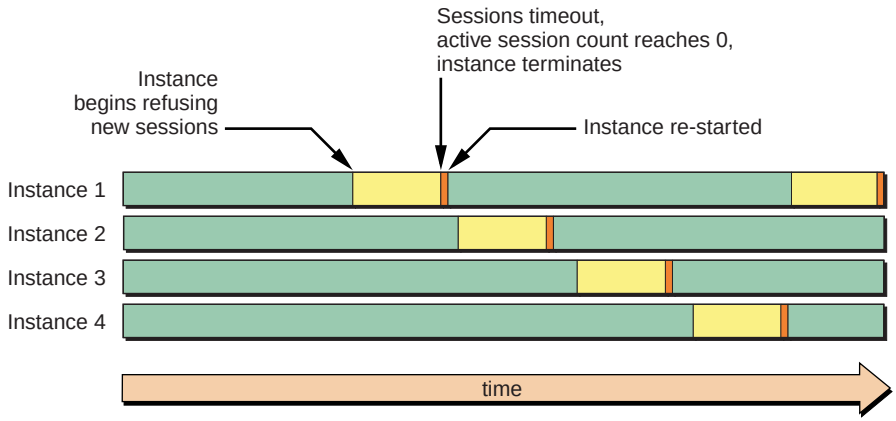
Automatic Scheduling

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 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:

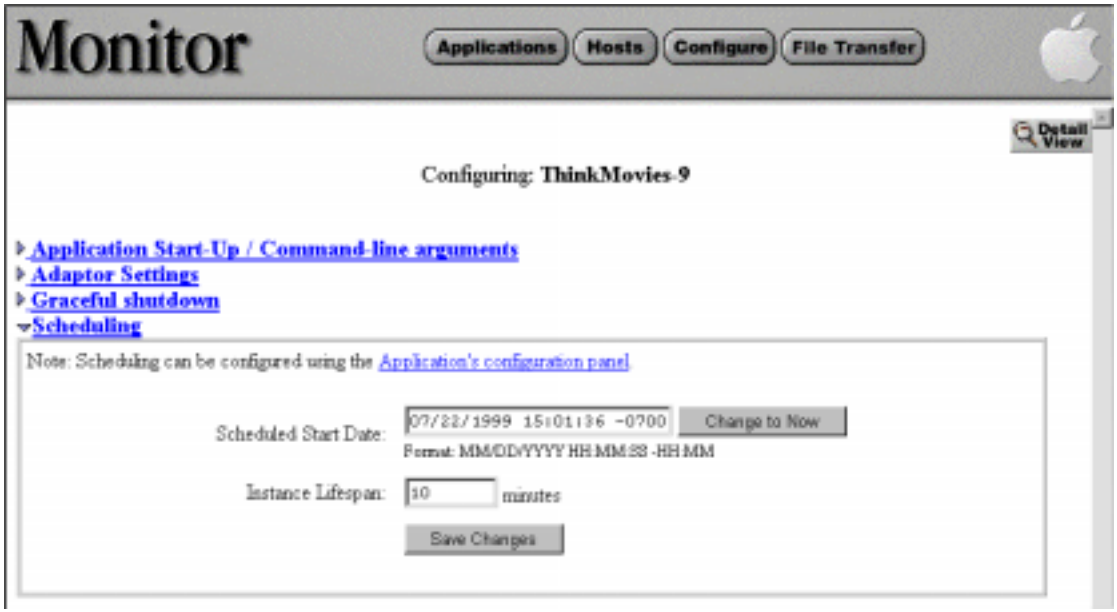


Either specify the instance lifespan or the frequency of shutdown (both in minutes) and then click the appropriate Use Option button. Each instance runs for a specified period before it begins refusing new sessions, and then it shuts down when the minimum active session threshold is reached. The diagram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



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
```

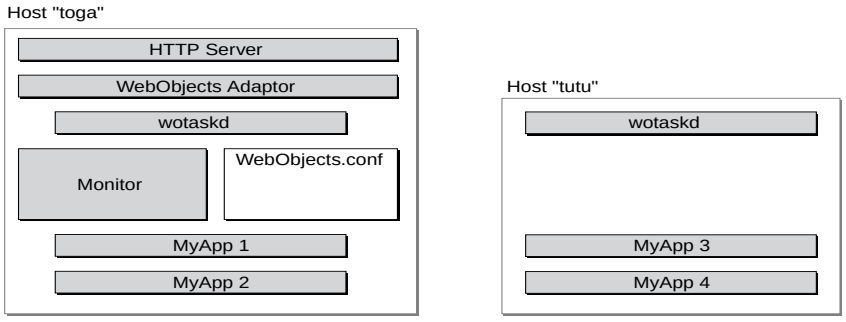
the WebObjects adaptor looks for an instance of MyApp on the host **toga**. If it finds an instance and the instance is ready to receive requests, the adaptor sends the request to that instance. If both of the instances of MyApp on **toga** are busy, it accesses an instance on the host **tutu**.

Use the Monitor application to create multiple new instances of an application for load balancing. See "Creating Application Instances" (page 29) for details.

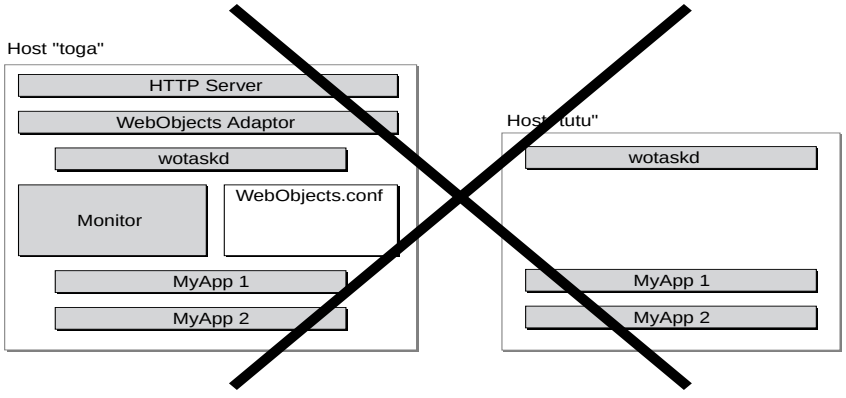
When you create multiple application instances, you are creating the configuration file ***NEXT_ROOT*/Library/WebObjects/Configuration/WebObjects.conf**. When the adaptor receives an HTTP request for an application, it first (in its initial mode) checks **WebObjects.conf** for an application instance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

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.

Even though instance numbers are unique across hosts, the web server adaptor's configuration file 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 web server adaptor's configuration 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** (see "Web Server Adaptor Configuration File Format" (page 11)). This is legal, but it's not supported by 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" (page 54)."
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

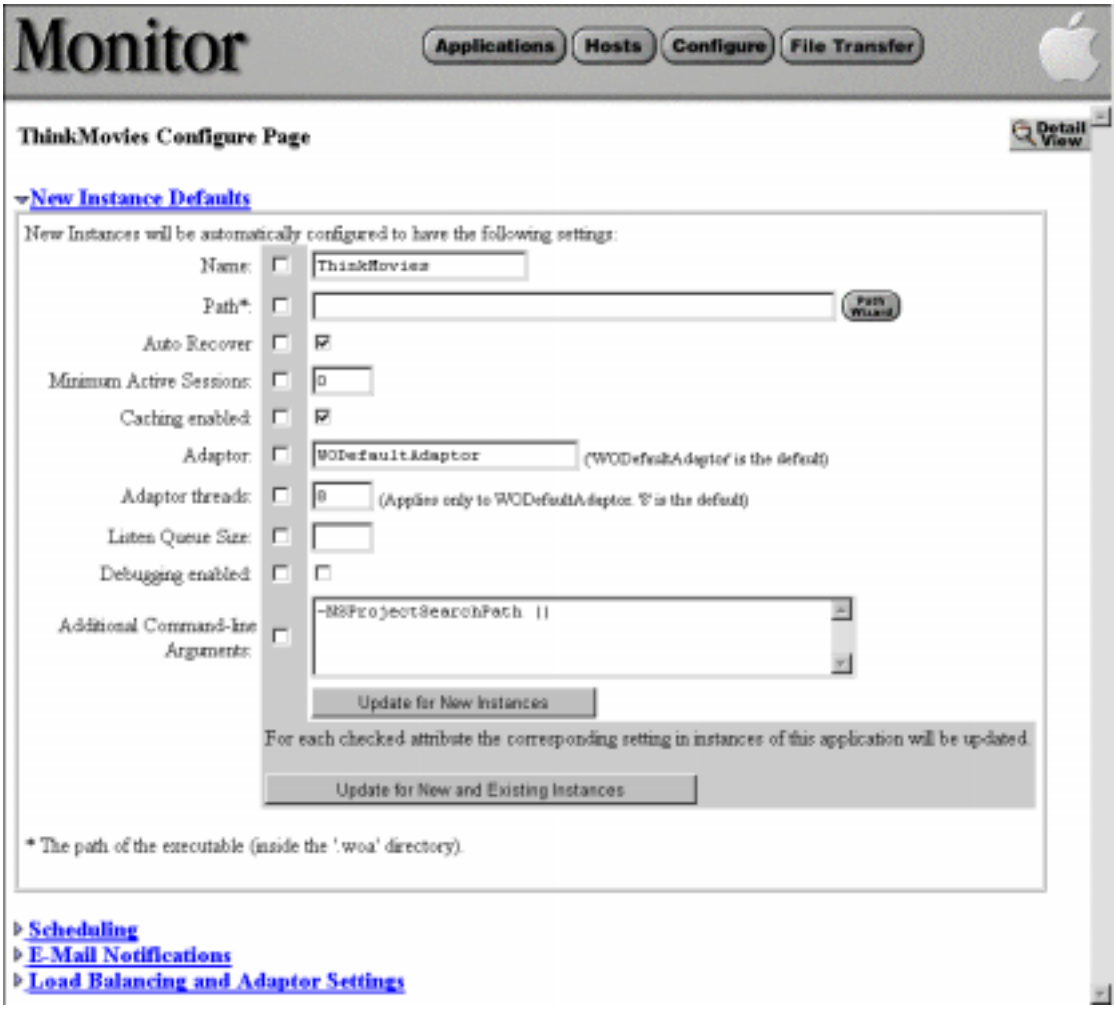
The listen queue depth indicates the number of transactions that can be in the socket buffer (the listen queue) awaiting processing. If the number of transactions in the buffer reach the limit set by the listen queue depth, the socket refuses new requests. The default depth is five.

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 New Instance Defaults option. This displays the following form:



4. Type the new listen queue depth in the Listen Queue Size field and check the box to the left of this 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 wotaskd Fail-safe

Because Monitor is a critical piece of any deployment, measures should be taken to make sure that it does not fail. As part of installation on both Windows NT and Mac OS X Server, **wotaskd** is configured to start automatically upon boot up. As well, it is started using **woservice**, which ensures that if **wotaskd** crashes for any reason it is automatically restarted.

Starting Monitor and wotaskd on Windows NT

During installation on Windows NT, **wotaskd** is configured to start automatically at boot time (in the Services control panel, it's listed as "Apple WebObjects Task Daemon"). If it doesn't appear to be starting correctly, check the Services control panel and ensure that **wotaskd**'s Startup mode is set to Automatic. As was previously noted, **wotaskd** is started under the control of **woservice**.

Monitor is also installed as a service (listed as "Apple WebObjects Monitor" in the Services control panel) and you can configure it to start automatically upon boot by changing its Startup mode to Automatic. Note that although this will cause Monitor to be started automatically, you'll have to manually start your web browser and connect to Monitor manually (or by putting a shortcut to your web browser in your Startup program group). The URL for Monitor is can be verified by checking the Windows NT Event Viewer (Start > Programs > Administrative Tools > Event Viewer) and is similar to:

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

Using woservice on Mac OS X Server

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

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

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

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

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

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

```
#!/bin/sh

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

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

The WebObjects Application URL

The typical WebObjects application URL has the following format:

`http://host[:port]/cgi-bin/WebObjects/App[[.woa][/instance]/key/...`

where the variables are defined as follows:

Variable	Description
host	The host name of your computer or **localhost**.
port	The port number. This is included if you want to direct connect.
cgi-bin	The cgi-bin directory of your server, usually **cgi-bin** or **Scripts**.
WebObjects	The name of the CGI adaptor, usually **WebObjects** or **WebObjects.exe**.
App	The application name. The **WOApplicationBaseURL** option provides the path to the application.
instance	The application instance number.
key	The request handler key. This key specifies which WORequestHandler object should be used to process the request., and is typically either “wo” (the component request handler) or “wa” the direct action request handler).
...	Information specific to the request handler. Each WORequestHandler uses a different format for the rest of the URL. The two main request handlers are WOComponentRequestHandler and WODirectActionRequestHandler. The WOComponentRequestHandler’s format for the rest of the URL is: *componentName*/*sessionID*/*elementID* WODirectActionRequestHandler handles direct actions. Its URLs have this format: [*actionClass*	*actionName* | *actionClass*/*actionName*][?*key*=*value*&*key*=*value*.....] For more detailed information, see the *WebObjects Developer’s Guide*.