WebObjects Deployment Guide Using JavaMonitor

Mac OS X Server > WebObjects

2007-10-31

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

Simultaneously published in the United States and Canada.

Contents

Figures, Tables, and Listings

Introduction to WebObjects Deployment Guide Using JavaMonitor

Note: This document was previously titled *WebObjects Deployment Guide*.

This document describes the tools and techniques that system administrators and website managers perform to deploy WebObjects applications. The WebObjects Deployment package allows you to deploy applications developed with the WebObjects Development package, so that they can be accessed through a Web server. You need a WebObjects deployment license to deploy WebObjects applications.

This document is intended primarily for system administrators. Application developers can also benefit from the information it provides but it's not required reading for them.

This document assumes you have a solid background in system administration. You must be familiar with the operation of your platform, especially how to use its command shell editor to issue commands. You must also be acquainted with the operation of your Web server software and TCP/IP networking. Knowledge of WebObjects application development is helpful, but not required.

To deploy WebObjects applications and to administer a deployment, you need to become acquainted with the deployment model of WebObjects. This document shows you how your Web server interacts with the elements of a WebObjects deployment. It also explains what measures you should take to increase your site's performance.

WebObjects Deployment provides tools for most of the tasks you need to accomplish on a regular basis to maintain your site. If you prefer doing things manually, you can use the command line to start individual application instances or the deployment tools themselves.

Organization of This Document

This document has the following chapters:

- "WebObjects Deployment" (page 13) gives you an overview of the deployment approach taken with WebObjects 5. In addition, it lists the ways in which WebObjects Deployment helps you to maintain a secure site.
- "Installing Software" (page 21) explains which WebObjects Deployment components need to be installed on a computer, taking into account the computer's purpose in your site. It also shows you how to build HTTP adaptors, which are interfaces between applications and web servers.

- "HTTP Adaptors" (page 29) describes the function of the HTTP adaptor in your site. It also describes how you customize the adaptors included in WebObjects Deployment if the default configuration does not suit your needs. State discovery is how the HTTP adaptor keeps track of the application instances of your site. The chapter describes the different ways that the adaptor can obtain that information and how you configure the adaptor to use one of those methods.
- "Managing Application Instances" (page 45) introduces you to the deployment tools you use to configure and maintain your site. It also describes the mechanism used in WebObjects to ensure that application instances are always running, helping you maximize your site's up time.
- "Deployment Tasks" (page 53) explains how to perform configuration and maintenance tasks on your site. It also shows how to maintain multiple sites using the same hardware.
- "Application Administration" (page 81) shows you how to monitor and improve your site's performance.
- "Application URLs" (page 89) describes the format of WebObjects application URLs so that you can use a web browser to connect directly to a running WebObjects application.
- "JMX Monitoring" (page 93) describes how to use JMX to monitor WebObjects applications.
- "Deployment Issues with Java Client" (page 99) lists issues to keep in mind when deploying Java Client applications. It also tells you what to do if you want to deploy WebObjects 4.5.1 applications together with WebObjects 5.x applications.

See Also

To get an overview of the WebObjects platform, you should read *WebObjects Overview*. You can find general information about WebObjects at http://developer.apple.com/webobjects.

Read *WebObjects Application Properties Reference* for a complete list of all the application properties for WebObjects applications, JavaMonitor, and wotaskd. These application properties correspond to command-line arguments that you can use to configure your WebObjects applications and tools for the particular deployment environment. Most of options in the JavaMonitor user interface correspond to JavaMonitor properties described in *WebObjects Application Properties Reference*.

WebObjects Deployment

This chapter introduces the essential concepts and tools you use when you deploy WebObjects applications.

The chapter contains the following sections:

- "The WebObjects Deployment Model" (page 13) introduces you to the WebObjects way of deploying applications. It explains how the users of your applications send requests to application instances running on your site and how responses (webpages) are generated and sent back to users.
- "The WebObjects Deployment Environment" (page 16) describes the functions of several elements (both in your platform and in WebObjects Deployment) in a site.
- "Keeping Your Site Secure" (page 20) lists the security-minded features available in WebObjects Deployment.

The WebObjects Deployment Model

A WebObjects deployment has six major parts:

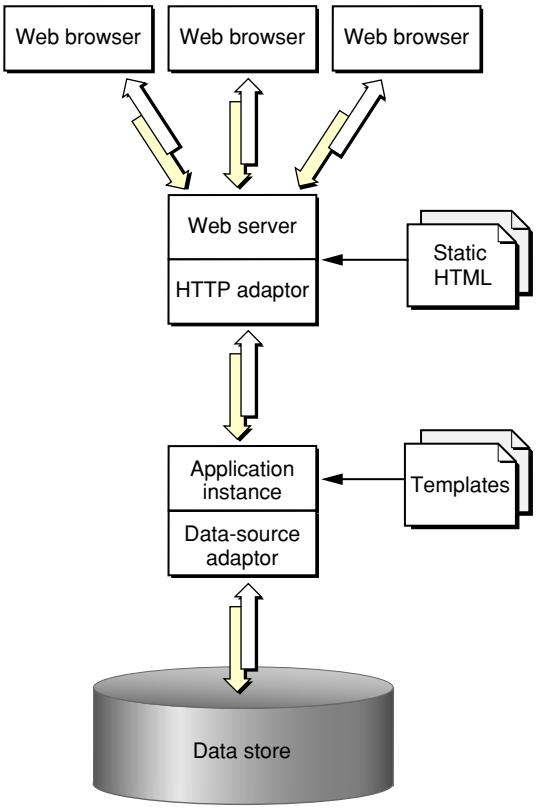
- **Client**: A user's web browser or the client side of a Java Client application.
- **Web server**: Application that receives HTTP requests from clients and sends responses back to them.
- **HTTP adaptor**: Application that serves as the interface between your web server and your application instances. The HTTP adaptor routes requests from the web server to the appropriate instance and sends the responses generated back to the web server. The adaptor does this while performing **load balancing** to distribute an application's users among its active instances. Load balancing helps to spread the user load of your site evenly across your application hosts.
- **Application instances**: Individual processes that receive requests from the HTTP adaptor and send responses back to it. To create a response, an instance can perform calculations, or save or retrieve data from a data store.
- **Data-store adaptor**: Interface between an application instance and a data store. WebObjects includes a **JDBC** adaptor, allowing your applications to connect to any JDBC-compliant data store. Also included is a JNDI (Java Naming and Directory Interface) adaptor, which allows applications to communicate with an LDAP (Lightweight Directory Access Protocol) server.

For JDBC connectivity, your database needs a JDBC driver, which you obtain from your database vendor. WebObjects applications can connect to databases that use Type 2 (partly Java) or Type 4 (all Java) JDBC drivers. The JDBC adaptor included with WebObjects Deployment has been certified to work adequately with Type 4 drivers. Type 2 drivers may require special configuration for them to work properly with the adaptor. If your database provides a Type 2 driver, consult with your database vendor to determine how it needs to be configured to work properly with a JDBC adaptor.

- **Data store**: The mechanism that your applications use to store persistent data. Consult with your database vendor or directory service vendor to obtain configuration and optimization details.

When an application user sends a request through a web browser to your web server, the server forwards the request to the HTTP adaptor. The adaptor then determines which application instance should process the request and forwards the request to it. When the application instance receives the request, it performs the necessary processing to produce a response (a new webpage). The instance then sends the response page to the adaptor, which forwards it to the web server. The web server then forwards the response page to the user's web browser. This process is illustrated in Figure 1-1.

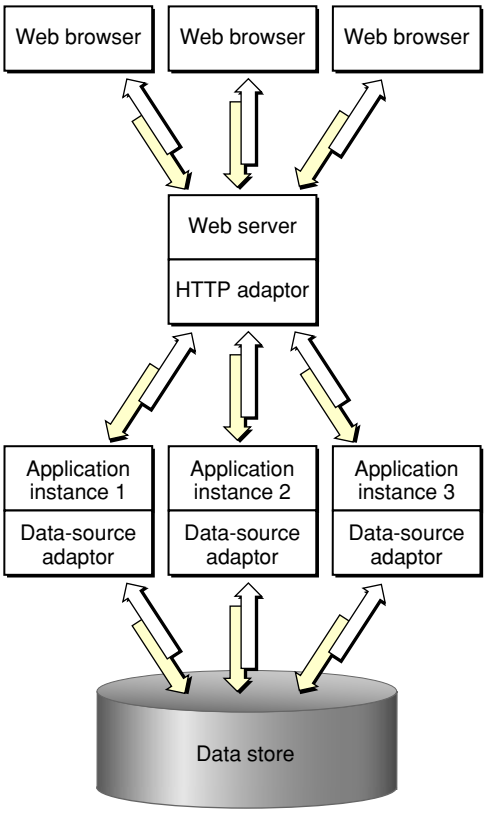
Figure 1-1 WebObjects deployment model



Notice that both the application instance and the web server contribute to the response page's content. The instance uses templates and logic to generate the HTML code for dynamic pages, while the web server provides the content of images contained in those pages. The server can also dispense static pages.

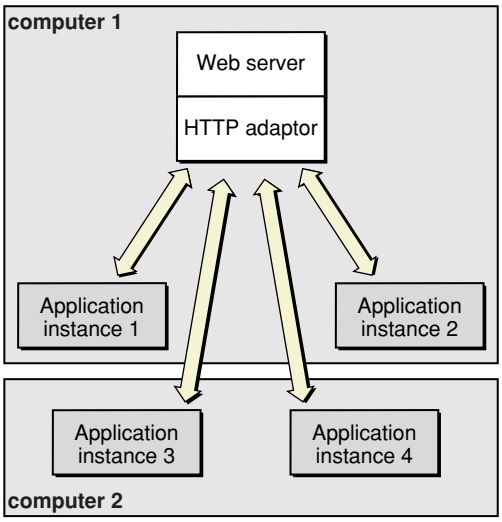
The number of instances of your application necessary to support its users depends on the number of users that connect to your application concurrently. In some cases a single instance is adequate. When one instance is not able to process requests in a timely manner, additional instances can solve the problem. This way, the amount of user-state information that a single instance stores is reduced. In addition, with less state to keep track of, an instance can process requests faster. Figure 1-2 shows a site with one host running multiple instances of an application.

Figure 1-2 WebObjects deployment model—multiple instances of an application



However, adding instances of your application to a host may not be the most effective solution. Eventually, a point of diminishing returns will be reached, where adding instances actually decreases your application's performance. In such a case, you should consider adding additional application hosts that run the extra instances required to handle the increased traffic to your site. Figure 1-3 shows how a site with two computers, one acting as an web server and application host, and the other just as an application host would look.

Figure 1-3 Deployment using two computers



The WebObjects Deployment Environment

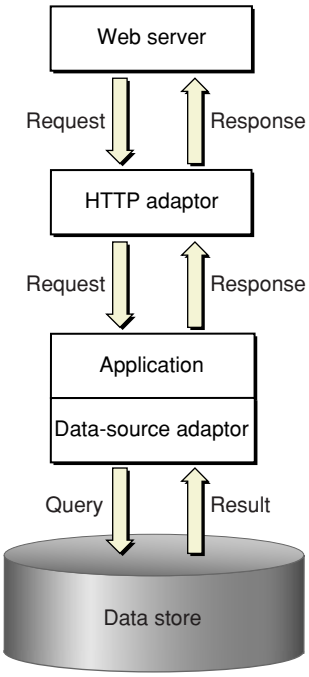
Before deploying applications, you need to master two important aspects of WebObjects Deployment: the communication paths of client requests and server activity, and the deployment tools you use to configure your site.

Communication Paths

Communication among the elements that make up a deployment occurs in two paths: the data path and the control path.

A client HTTP request takes the data path after it reaches your web server. Figure 1-4 shows how an HTTP request that your web server receives is passed to the elements that generate the response.

Figure 1-4 The data path of a WebObjects deployment



JavaMonitor requests take the control path to propagate configuration changes to application hosts and, ultimately, application instances. These include adding application instances and starting and stopping instances according to a schedule that you define. The HTTP adaptor can obtain site information by polling wotaskd (WebObjects task daemon) processes or by reading the adaptor configuration file. (See "Deployment Tools" (page 18) for information about JavaMonitor and wotaskd.) Figure 1-5 shows the control path.

Figure 1-5 The control path of a WebObjects deployment

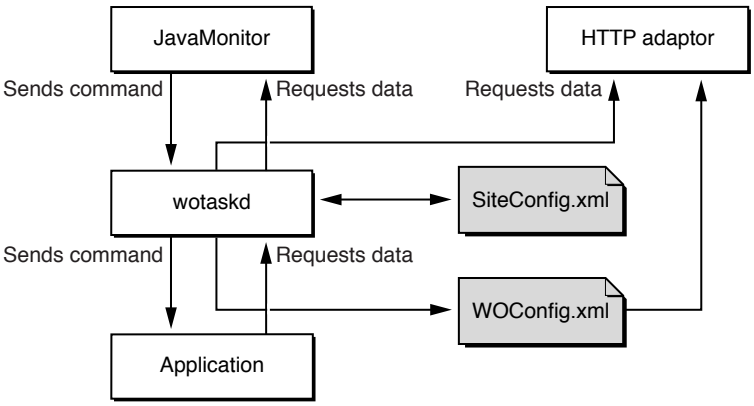


Figure 1-6 shows how the data path and control path are differentiated in the rest of the document.

Figure 1-6 The symbols used to represent the data path and the control path

Deployment Tools

The main tools you use to manage your site are wotaskd and JavaMonitor. Normally, one wotaskd process runs on each application host. If you want to concurrently deploy multiple sites on the same hardware, you can configure a computer to run more than one wotaskd process. This essentially provides you with several independent application hosts per computer.

You manage a group of application hosts using JavaMonitor, a tool that uses your web browser as its user interface. JavaMonitor lets you set, among other things, instance scheduling and the load-balancing scheme to be used for each application. Because each JavaMonitor process maintains state information locally, you must run only one instance of JavaMonitor per site. Figure 1-7 shows two application sites on one computer.

Figure 1-7 Two sites deployed on one computer

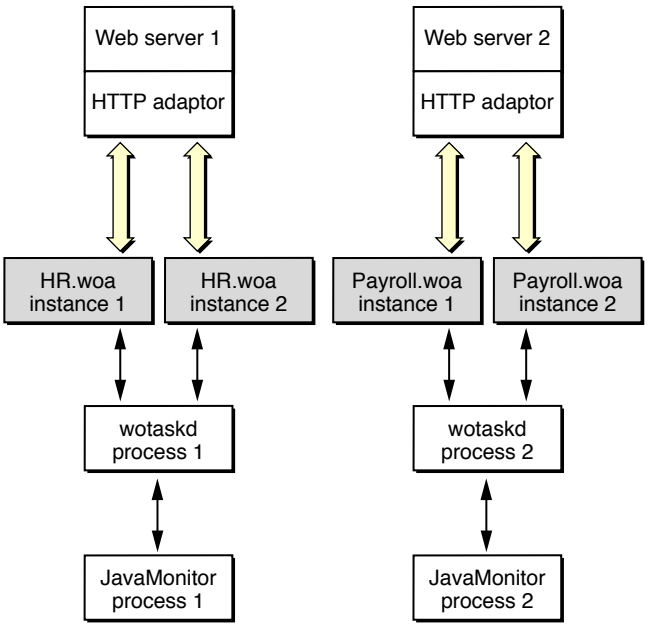
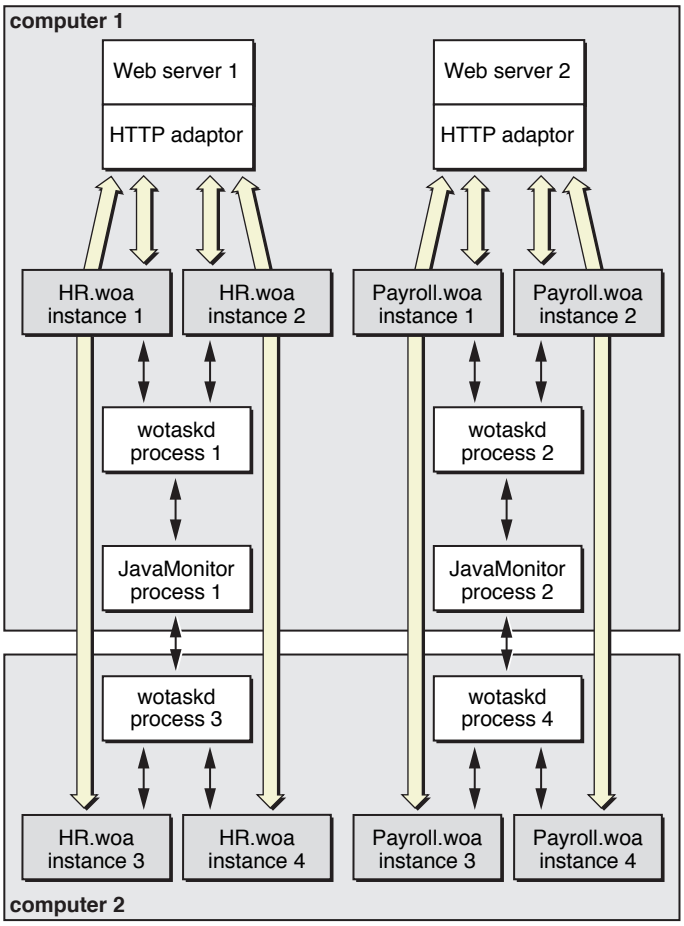


Figure 1-8 shows how you can distribute application instances among two computers.

Figure 1-8 Two sites deployed on two computers



After you configure your site using JavaMonitor, it enforces that configuration by performing tasks such as stopping and restarting application instances according to a schedule you set and sending email notifications when problems arise. The HTTP adaptor performs load balancing across the instances of each application on your site.

For detailed information on the subjects introduced above, see the following chapters or sections:

- "HTTP Adaptors" (page 29) shows you the different ways in which you can configure the WebObjects HTTP adaptor.
- "Deployment Tasks" (page 53) describes how you use JavaMonitor to configure your site.
- "Setting Up Hosts" (page 53) describes how you use JavaMonitor to add application hosts to your site.
- "Configuration Files" (page 45) shows you how the configuration you define in JavaMonitor is distributed among the application hosts of your site.
- "wotaskd Processes" (page 47) explains how wotaskd processes communicate with and manage application instances.
- "Lifebeats" (page 46) explains how application instances communicate with a wotaskd process.

- "Deploying Multiple Sites" (page 78) explains how to configure your platform to deploy multiple sites concurrently.
- "Load Balancing" (page 77) describes how load balancing works and lists the algorithms that the HTTP adaptor can use to implement it.

Keeping Your Site Secure

In a WebObjects deployment, you have several features at your disposal to enhance the security of your site:

- **split-installation of applications (application files and web server resources)**: By installing application-related files in two locations, you can put sensitive information (such as business logic) into protected locations. Nonsensitive resources (such as image files) can be installed on the Web server's `Document Root` directory. For more information, see "Installing Applications" (page 58).
- **restricted access to deployment tools**: "JavaMonitor Password" (page 76) explains how you can password-protect access to JavaMonitor and wotaskd through a single page.
- **restricted access to development application instances**: If your computing environment supports both the development and deployment of applications through the same Web server, access to development instances is restricted by the HTTP adaptor. See "Viewing a Host's Configuration" (page 56) for details.
- **ability to disallow direct connections (through host name and port number) to an application instance**: With direct connect you can connect to an instance with the following URL:

  ```
  http://myhost:1234
  ```

  When you disallow direct connect for an instance, the only way to connect to it is through an Web server. For more information, see `WODirectConnectEnabled` in *WebObjects Application Properties Reference*.
- **restricted access to application-instance statistics**: Agents external to your organization can use the statistics that your application instances produce to get privileged information. To avoid this, access to the instance statistics page is restricted. See "Setting a Password for the Instance Statistics Page" (page 74) for details.
- **restricted access to HTTP adaptor information (the WebObjects Adaptor Information page) by default**: This closes another potential hacker entry point. For details, see "Setting Access to the WebObjects Adaptor Information Page" (page 43).
- **restricted access to the wotaskd port**: wotaskd uses Port 1085. The access to this port (both UPD and TCP) must be protected through a firewall in order to have a secure environment. See also "Security Issues with wotaskd" (page 48)

Installing Software

Important: In WebObjects 5.3 and later, you do not need a deployment license key. WebObjects Deployment is preinstalled with Mac OS X Server 10.4.1 and later. However, installing earlier WebObjects deployment packages requires a deployment license key. Read the license agreement before installing earlier releases.

Important: On Mac OS X Server v10.5 and later, you have the option of either keeping the configuration files for Apache 1.3 and continuing to use Apache 1.3, or upgrading to Apache 2. Once you upgrade to Apache 2, you cannot return to Apache 1.3.

A clean install of Mac OS X Server v10.5 will always use Apache 2.2. If you wish to keep Apache 1.3, you should do an upgrade install instead of a clean install.

The Migration Assistant automatically converts your Apache 1.3 configuration files to Apache 2 when you upgrade to Apache 2. After an upgrade install, if you want to upgrade your Apache Version from 1.3 to 2.2, start the Server Admin application and select the Web module. Select "Upgrade Apache Version" to upgrade to Apache 2. If there is a problem during the translation process, the conversion stops and your Apache 1.3 configuration files are left unmodified. Errors, warnings, or confirmations of successful operations during the conversion are logged at `/Library/Logs/Migration`.

See *Mac OS X Server Upgrading and Migrating* and *Mac OS X Server Web Technologies Administration*for more details.

Important: If your WebObjects adaptor does not start properly with the Apache 2.2 module, then check if the Apache configuration file `httpd.conf` is missing a reference to the WebObjects adaptor.

To do this, open the `/etc/apache2/httpd.conf` file and search for this line:

```
Include /System/Library/WebObjects/Adaptors/Apache2.2/apache.conf
```

If this line does not exist, add these lines to the `/etc/httpd/httpd.conf` file and restart your web server:

```
# Including WebObjects Configs
Include /System/Library/WebObjects/Adaptors/Apache2.2/apache.conf
```

This chapter addresses WebObjects Deployment installation issues, including what elements of the software need to be installed on a computer, taking the computer's purpose into account.

The chapter contains these sections:

- "Choosing What to Install" (page 22) explains which files need to be installed on a computer, based on the purpose of the computer within a deployment.
- "HTTP Adaptors" (page 23) covers the various HTTP adaptors that can be used in a deployment. It explains which adaptors are appropriate for your platform as well as how to build an adaptor from the source files included with WebObjects.
- "Data-Store Adaptor Installation" (page 27) describes the JDBC and JNDI adaptors that WebObjects applications can use to interact with data stores, such as databases and directory-services servers.

Choosing What to Install

When you perform a complete installation of WebObjects Deployment on a computer, two types of files are copied to its hard disk: adaptor files and deployment files.

HTTP Adaptor Files

HTTP adaptors allow your web server to communicate with WebObjects application instances. The adaptor processes a single transaction, a request and its response, at a time. Normally, the request is forwarded to a single application instance, which, in turn, generates the response. Exceptions include error conditions or a request for an application that does not exist.

Typically an HTTP adaptor performs the following actions for each request:

1. Reads the request from the server, checks the URL, and collects header data and form data.
2. Finds an application to service the request. This is the part of the process that involves load balancing between instances.
3. Uses an existing socket connection to the application or connects a new socket to the application and forwards the request to the application.
4. Waits for and reads the response (status, headers, and content).
5. Returns the connection to the connection pool or—if the connection is transitory—closes the connection.
6. Sends the response to the client through the web server.

The executable files of the HTTP adaptors are placed in `/System/Library/WebObjects/Adaptors`. You can build your own adaptors using the source files for the default adaptors as a starting point. These source files are installed in `/Developer/Examples/WebObjects/Source/Adaptors`.

In addition to the HTTP adaptors, WebObjects applications that access a data store need to have an appropriate adaptor for that data store. Two types of data stores are supported, databases and directory services; these use JDBC and JNDI (Java Naming and Directory Interface), respectively.

Deployment Files

Deployment files are divided into two groups:

- **Runtime environment**. The runtime environment of WebObjects is implemented in JAR (Java archive) files contained within **framework** (`.framework`) bundles. WebObjects frameworks are installed in `/System/Library/Frameworks`. These frameworks are used by any WebObjects application, including the deployment tools of WebObjects.
- **Deployment tools**. WebObjects Deployment includes two deployment tools you use to configure and monitor your site. The files that make up these tools are placed in `/System/Library/WebObjects/JavaApplications`. Table 2-1 shows the purpose of each tool. For more information on the deployment tools, see "Managing Application Instances" (page 45).

Table 2-1 The WebObjects deployment and administration tools

Filename	Application name	Purpose
`JavaMonitor.woa`	JavaMonitor	Site configuration and administration
`wotaskd.woa`	wotaskd	Instance management

Note: In WebObjects 5.4, the source for the deployment tools is located in the `/Developer/Examples/JavaWebObjects/Source/JavaMonitor` folder.

Types of WebObjects Deployment Installations

Depending on the purpose of the computer you're installing the software on, there are three types of WebObjects Deployment installations you can perform:

- **Web server only**: On a computer that you want to use as the web server but on which you do not intend to run application instances (including the deployment tools), you need to install only the HTTP adaptor files.
- **Application host only**: On computers that you intend to use only as application hosts, you need to install only the deployment tools. (If you do not plan to run JavaMonitor on that computer, you can delete its files.)
- **Web server and application host**: When one computer can satisfy all your deployment needs or when you want a web server to also run application instances, you need to install the adaptor files and the deployment tools.

HTTP Adaptors

WebObjects uses a variety of HTTP adaptors to enable access to applications through a web server. Depending on your deployment platform, particular HTTP adaptors are installed by default.

Mac OS X Server supports Apache and CGI. The Apache adaptor is active by default. Requests in the form `http://.../cgi-bin/WebObjects/` are handled by the Apache adaptor. If you disable the Apache module, such requests are handled by the CGI adaptor.

General Adaptor Building Issues

Important: In WebObjects 5.4 and later, recompiling the Apache 1.3 adaptor is not supported. The source for the Apache 2 adaptor is located in the `/Developer/Examples/WebObjects/Source/Adaptors/Apache 2` folder in the WebObjects development package.

If you want to use an HTTP adaptor other than the one installed by default on your platform, you need to build it and install it. This section discusses building HTTP adaptors from the source code in `/Developer/Examples/WebObjects/Source/Adaptors`.

Before building an adaptor, make sure that you have the following installed:

- WebObjects Deployment
- ANSI-C compliant compiler—for example, `gcc` version 2.7.2 or later
- `gnumake` version 3.74 or later
- Web server software

Follow these steps to build your adaptor:

1. Select the platform you want to build on by editing the `ADAPTOR_OS` variable in `make.config`.
2. Select the adaptor that you want to build by editing the `ADAPTORS` variable in `make.config`. You can build multiple adaptors for different web servers at the same time.
3. Modify the appropriate compile-time parameters in `Adaptor/config.h`. Most features can be configured at initialization time. Some, however, are determined during compilation; you change these by editing `config.h`.
4. Depending on the adaptor you want to build, you may need to make additional changes:
    - For Apache, modify the `APXS` variable in `make.config`. The default settings should work for Mac OS X Server.
    - For CGI, you might need to customize `CFLAGS` or `LDFLAGS` in `CGI/Makefile`. For example, you don't need to define `-nopdolib` when you are not using the Apple PDO C compiler.
5. Set the `CC` variable in `make.config` to point at the compiler you want to use. The compiler must be an ANSI C compiler. The `CC` variable is set to `gcc` by default.
6. Type `make` in the `NEXT_ROOT/Developer/Examples/WebObjects/Source/Adaptors` folder.

Note: To compile the Apache 2 adaptor on Mac OS X, type `make Apache2.2`.

In case you need to modify the default adaptors, Table 2-2 identifies the purpose of the source files in `/Developer/Examples/WebObjects/Source/Adaptors/`.

Table 2-2 Adaptor source description

Source file	Description
`Apache/mod_WebObjects.c`	Interface modules that interact directly with the web server.
`CGI/WebObjects.c`	Interface modules that interact directly with the web server.
`Adaptor/config.h`	Configuration-parameters file.
`Adaptor/config.c`	Operating system–specific configuration items.
`Adaptor/transaction.c`	Implements request/response processing.
`Adaptor/request.c`	Structures and routines to collect headers and form data. Also function support for sending requests to applications.
`Adaptor/response.c`	Structures and routines to collect response headers and content.
`Adaptor/hostlookup.c`	Gets `hostent` structure for an application host using `gethostbyname` and caches the result.
`Adaptor/transport.h`	Declaration of application IPC API.
`Adaptor/nbsocket.c`	Transport is implemented with nonblocking sockets. This file provides timeouts and user-space buffering.
`Adaptor/loadbalancing.h`	Declaration of functions to provide load balancing. Load-balancing implementations need to define these functions.
`Adaptor/random.c`	Load-balancing routine that randomly selects any available application instance.
`Adaptor/roundrobin.c`	Load-balancing routine that selects an instance using a round-robin algorithm.
`Adaptor/loadaverage.c`	Load-balancing routine that selects an instance using the load average returned by each instance in its headers.
`Adaptor/log.c`	`printf` style logging to `/tmp/WebObjects.log`. Checks for the existence of `/tmp/logWebObjects`, an empty file that must be owned by root on UNIX platforms. This file should not be present in deployment mode.
`Adaptor/xmlparse.c`	Parses the configuration information supplied in an XML document. This is either supplied from wotaskd, a URL, or a file.
`Adaptor/WOURLCUtilities.c`	WebObjects URL-parsing routines.
`Adaptor/MoreURLCUtilities.c`	Additional utility functions to augment `WOURLCUtilities.c`.
`Adaptor/strdict.c`	String key–based dictionary.

Source file	Description
Adaptor/strtbl.c	String key/value lookup data structure.
Adaptor/list.c	Data structure used to collect pointers.

Building the Apache Adaptor

A successful build of the Apache adaptor yields a file named mod_WebObjects.so. This file should be copied into /System/Library/WebObjects/Adaptors/Apache on Mac OS X and NEXT_ROOT/Library/WebObjects/Adaptors/Apache on other platforms. In order for the adaptor to work, Apache must be configured to accept Dynamic Shared Objects (DSOs). Refer to the Apache web server documentation available at http://www.apache.org for more information on building Apache to accept DSOs. If the adaptor fails to build, it is probably because Apache is not built to accept DSOs.

Once you have built the adaptor and server, you need to configure the web server to handle WebObjects requests. See "Customizing HTTP Adaptors" (page 41) for more information on configuring Apache.

After you have built and configured the server with the linked adaptor, you should start it and confirm that it's working by moving aside the WebObjects CGI adaptor in the cgi-bin directory and making a few requests. You can determine whether the CGI or Apache adaptor is handling requests by turning on the logging feature of the adaptor as follows:

1. As root, execute the following command:

   ```
   touch /tmp/logWebObjects
   ```

2. Make a request to a WebObjects application to initialize the log file.

3. Execute the following command:

   ```
   tail -f /tmp/WebObjects.log
   ```

4. If the Apache web server is configured to use the CGI adaptor, each request is logged as

   ```
   Info: <CGI> new request: /cgi-bin/WebObjects/MyApp
   ```

5. If the Apache web server is configured to use the WebObjects Apache module, each request is logged as

   ```
   Info: <WebObjects Apache Module> new request: /cgi-bin/WebObjects/MyApp
   ```

Building the CGI Adaptor

The default CGI adaptor is a generic CGI adaptor designed to be used with all web servers that support CGI. There is a performance disadvantage when using CGI adaptors; therefore, you should consider using a server plug-in adaptor whenever possible.

To install this adaptor, copy the file WebObjects to your web server's cgi-bin or scripts directory. This is done for you if you install WebObjects on a system with a web server installed.

It is possible to configure the CGI adaptor to contact the instance of wotaskd on localhost for adaptor configuration information including the list of instances. For deployment you normally want to use a different mechanism that is less expensive. Set up the CGI adaptor to use either a static file on the web server or a static URL for this information. For examples of this, see "Customizing HTTP Adaptors" (page 41).

If there are problems executing the CGI adaptor on MacOS X, do the following:

- Make sure that the `WebObjects` CGI executable is installed in `/Library/WebServer/CGI-Executables/`.
- Verify that it is owned by root:admin.
- Make it executable by everyone.

Although generally not the best for production-level deployment, there is a good reason to use the CGI adaptor. It is useful for exercising the underlying request handler and debugging any customizations you may have made to the source code. Since all input to any CGI program is provided in the environment variables and `stdin`, the WebObjects CGI program can be conveniently run under a debugger.

To do this, set the following environment variables and values as shown in Table 2-3.

Table 2-3 CGI debugging variable settings

Environment variable	Value
`REQUEST_METHOD`	`GET`
`SERVER_PROTOCOL`	`HTTP/1.0`
`QUERY_STRING`	`\?foo=bar`
`SCRIPT_NAME`	`/cgi-bin/WebObjects`
`PATH_INFO`	`/MyApps/MyCoolApp`

If you want to include form data, set a `CONTENT_LENGTH` header and type the form as `stdin`. An alternative is to edit and execute the `TestCGI.sh` or `Env.csh` files provided in `/Developer/Examples/WebObjects/Source/Adaptors/CGI`.

Data-Store Adaptor Installation

This section provides resources that help you obtain and install the two types of data-store adaptors that WebObjects applications may require: database adaptors and adaptors for directory-services servers.

Database Adaptors

To ensure that your JDBC driver works with your database, you should get the latest JDBC driver that matches the version of your database and the version of Java JDK installed on your machine.

Directory Services Adaptors

WebObjects provides access to directory services through the use of the JNDI adaptor in conjunction with a service provider. WebObjects has been tested with connections to the following Lightweight Directory Access Protocol (LDAP) servers:

- OpenLDAP Directory Server

To use this LDAP JNDI adaptor, you need to have the JNDI class libraries and the LDAP service provider from Sun Microsystems installed. These are both available from http://java.sun.com/products/jndi. On Mac OS X these are installed by default.

HTTP Adaptors

This chapter provides detailed information about the HTTP adaptors that are included in a WebObjects Deployment installation. The HTTP adaptor is an important piece of an application site. It sits between your web server and your application instances. It forwards requests from the web server to the appropriate application instance and responses from the instance back to the web server. It also performs load balancing between instances of an application.

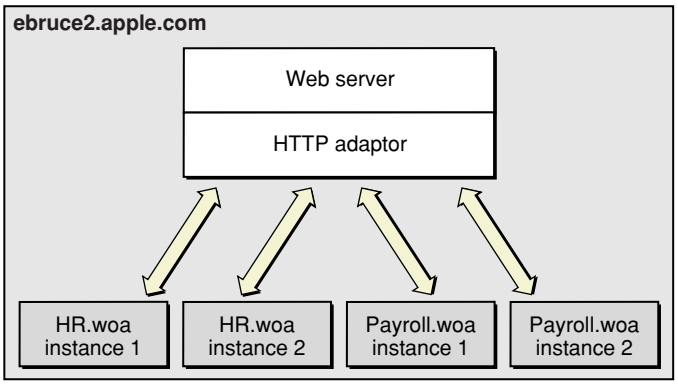
This chapter contains the following sections:

- "Adaptors, Applications, and Hosts" (page 29) provides a high-level view of the interaction between the HTTP adaptor, application instances, and application hosts.
- "Types of Adaptors" (page 31) explains the differences among the two types of HTTP adaptors that you can use on your site.
- "State Discovery" (page 32) describes how to configure an HTTP adaptor to obtain your site's state dynamically or using a configuration file. It also explains how to change a dynamic configuration into a static one.
- "The WebObjects Adaptor Information Page" (page 40) shows an example of the web browser page that displays information about an HTTP adaptor.
- "Customizing HTTP Adaptors" (page 41) summarizes all the settings available for HTTP adaptors.

Adaptors, Applications, and Hosts

The HTTP adaptor forwards requests from an web server to application instances and returns responses from instances back to the server. You may need to have more than one instance of a given application to support a large number of concurrent users. Figure 3-1 illustrates a simple site, implemented with one computer. It serves two applications, with two instances for each application.

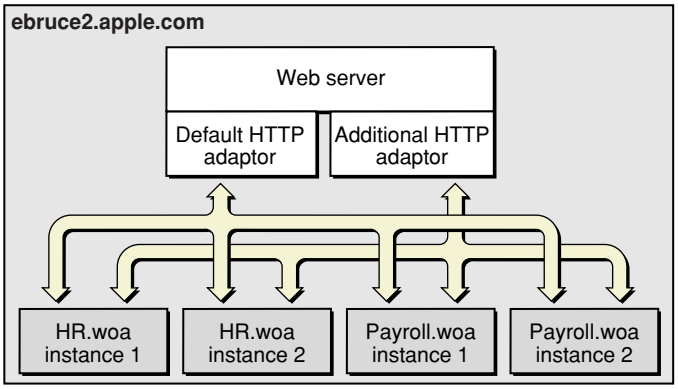
Figure 3-1 Deployment on one computer, using one adaptor



Although the WebObjects installation provides several HTTP adaptors, only one is active by default (see "HTTP Adaptors" (page 23) for details). However, an application instance can communicate with an adaptor other than the active adaptor.

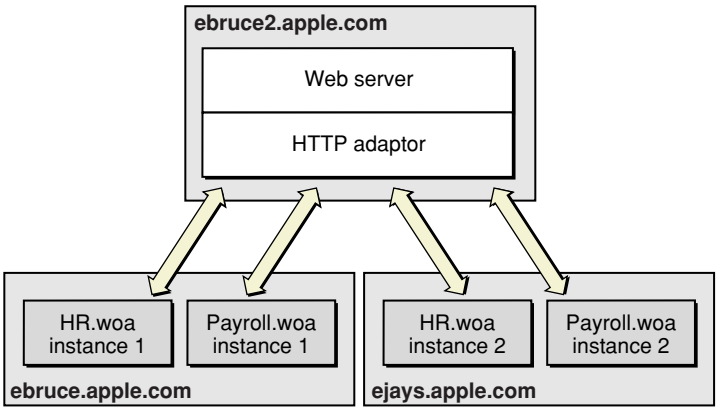
Figure 3-2 depicts an application site running on one computer, using two adaptors.

Figure 3-2 Deployment on one computer using two adaptors



Most sites require multiple computers to ensure that an instance of a particular application is always available. In this kind of deployment, usually one computer runs the web server and the HTTP adaptor, while one or more additional computers serve as application hosts. Figure 3-3 illustrates an application site using three computers, one running the web server and the adaptor, and the other two running application instances.

Figure 3-3 Deployment using three computers using one adaptor



The HTTP adaptor needs to periodically determine your site's state—which application instances are running. There are two ways in which the adaptor can obtain this information:

- **dynamically**: The adaptor determines your site's state by asking each application host for its state. The adaptor can use a multicast request to find out which hosts are available or you can define a host list for it. Using this method, you avoid having to configure new hosts as you add them to your site.
- **from a static file**: An adaptor configuration file contains host and application information about your site; it includes information about every application instance you want to run. After the adaptor reads the file, it has all the information it needs to communicate with the application instances you want to run. Using this kind of configuration avoids multicast requests and host polling. However, when you add new hosts, you have to update the configuration file. For more information on the adaptor configuration file, see "Using a Configuration File" (page 35).

Types of Adaptors

There are two general types of HTTP adaptors, **CGI adaptors** and **API-based adaptors**. CGI adaptors are portable across many platforms. API-based adaptors are generally more efficient than CGI adaptors.

CGI Adaptors

WebObjects Deployment includes a CGI adaptor, which is an executable file named `WebObjects`. The CGI adaptor resides in the Web server's `cgi-bin` or `scripts` directory. This adaptor works with any web server that conforms to the CGI standard.

The major drawback of CGI adaptors is their performance. When the web server receives a request from a web browser, it creates a new process for the HTTP adaptor. When the adaptor is done processing the request, the process is terminated.

The CGI adaptor is installed by default on all platforms, but it may not be the active one on your platform. See "HTTP Adaptors" (page 23) for more information.

API-Based Adaptors

API-based adaptors are founded on API specific to a particular web server. They allow CGI-like tasks to run as part of the main server process, avoiding the creation and termination of a process for each request. WebObjects Deployment includes Apache's module API.

State Discovery

Your site's state is represented by

- a list of application hosts
- a list of running application instances on each host

The HTTP adaptor captures your site's state at regular intervals, which you set when you configure the adaptor. You also define the method that the adaptor uses to gather state information by configuring the adaptor itself. For details, see "Customizing HTTP Adaptors" (page 41).

The adaptor can obtain the state of your site using one of three methods:

- **Multicast request**: The adaptor sends a multicast request to find out what application hosts are available. After compiling the host list, the adaptor polls each host to get its list of running application instances.
- **Static host list**: This method requires that you configure the host list in the adaptor itself. As with the first method, the adaptor polls the hosts on the list for their lists of running application instances.
- **Configuration file**: The adaptor obtains the site's configuration by reading an **XML** (Extensible Markup Language) document.

The method that requires the least administration on your part is the multicast request. If an application host goes down, the adaptor automatically removes the application instances running on it from its list of active instances. When the host is brought back up, the adaptor adds the instances back to its list. You should use this method if your site has many application hosts. See "Using a Multicast Request" (page 33) for more information.

The second method, defining a host list for your adaptor, eliminates the multicast request. Use this method if you do not want the adaptor to send regular multicast requests out on your network or if you seldom add or remove application hosts from your site. This is the method that is active by default. However, the host list contains only one host, `localhost`. For details, see "Using a Static Host List" (page 35).

In the third method, using a configuration file, the HTTP adaptor obtains your site's configuration by reading a file. This file can be static or it can be dynamically updated as you configure your site with Monitor. For details, see "Using a Configuration File" (page 35).

You can write the adaptor configuration file in one of two ways:

- **Manually**: The information in the configuration file is stored in an XML document. For details, see "The HTTP Adaptor Configuration File" (page 35).

- **Using JavaMonitor and wotaskd**: After configuring your site to your liking using JavaMonitor, you can have a file created for you or you can copy-and-paste the information. See "Creating the HTTP Adaptor Configuration File" (page 37) for more information.

Using a Multicast Request

When you configure an adaptor to obtain your site's state using a multicast discovery request, the adaptor obtains the list of active application hosts by broadcasting a message to which each computer configured as a WebObjects application host responds. After the adaptor compiles the list of available hosts, it polls each one to obtain its state (the list of running application instances).

There are drawbacks to using the multicast method:

- It increases network traffic. By default, the HTTP adaptor sends a multicast request every 100 seconds.
- A host may become unavailable between discovery requests if the multicast request or a wotaskd process's response is lost (multicast is an inherently unreliable protocol).
- Normally, multicast broadcasts are limited to a subnet. However, you can configure your routers to pass-on the multicast request to other subnets if you wish.

Warning: By default, wotaskd does not respond to multicast requests. To be able to use the multicast request method, you must configure wotaskd processes in your application hosts to respond to multicast requests. If you configure wotaskd to respond to multicast requests, your server may be vulnerable to attacks unless properly protected by a firewall or other security measures.

Multicast Request

To discover available hosts, the adaptor sends a host-discovery request on the multicast channel (a nonrouting IP address and a port number), which is set to IP address `239.128.14.2` and port `1085` by default. The frequency of each multicast request is ten times as long as the adaptor's configuration refresh interval. For details on how to change the multicast channel, read "Setting the Multicast Address and Port" (page 42). Also, see `WOPort`, `WOMulticastAddress`, and `WORespondsToMulticastQuery` in *WebObjects Application Properties Reference*. When a wotaskd process starts, it creates a UDP (User Datagram Protocol) socket that listens to the multicast channel through which it receives multicast requests.

Note: If you change the address and port that adaptors use to send multicast requests, you must also change the address and port that wotaskd processes use to receive multicast requests.

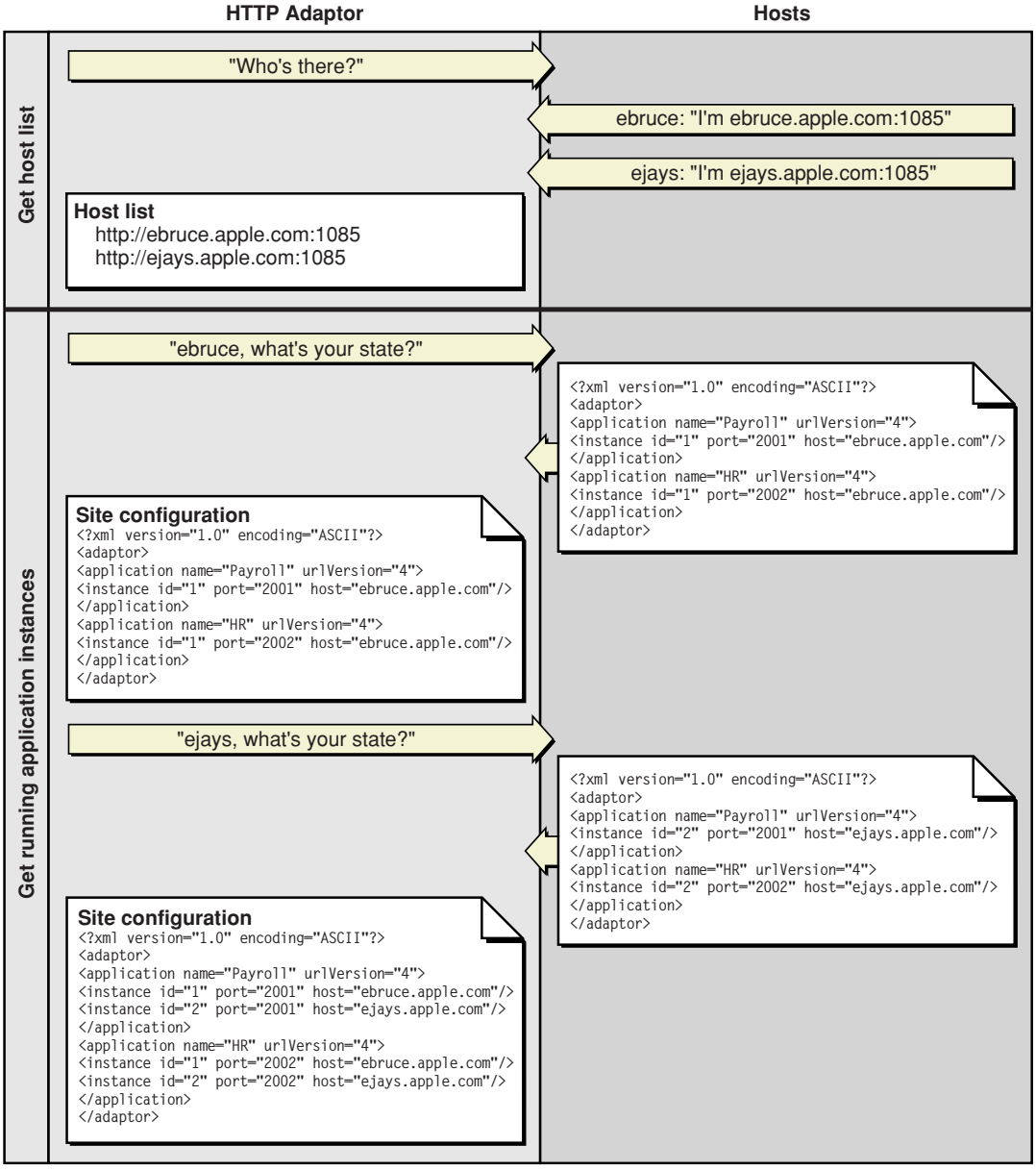
When each wotaskd process receives the multicast request, it replies with its URL, such as `http://host1.site.com:1085`. The adaptor in turn compiles a list of these URLs.

Sending a multicast request on an entire subnet is an expensive process. If your available hosts never change, consider using a static host list instead.

Host Polling

After the HTTP adaptor constructs the host list, it polls each application host on the list for information on the active application instances running on it. Each wotaskd process, in turn, sends its state information using the format in Listing 3-2 (page 36). Host polling to obtain information on active instances occurs at the interval indicated in the configuration refresh interval setting for the HTTP adaptor. Figure 3-4 illustrates the process used to determine the configuration of the site in Figure 3-3 (page 31).

Figure 3-4 Dynamic site configuration using multicast request and polling



Using a Static Host List

This method is similar to the one described in "Using a Multicast Request" (page 33). The only difference is that the HTTP adaptor skips the first part, the multicast request. The host polling process occurs at the interval set in the adaptor's configuration refresh interval setting.

You must explicitly define a host list for each adaptor. See "Setting the Host List" (page 42) for details on defining the host list for each of the adaptors provided.

Using a Configuration File

Using an HTTP adaptor configuration file is useful when you want to have a static site configuration (one in which application instances are not stopped after they are started) or if you want to use JavaMonitor to configure your site and have the adaptor read your configuration changes immediately. (The adaptor reads the configuration file every 10 seconds to determine which application instances are active.)

This method also provides a way of having more than one configuration of your site available. You can switch among different configurations by placing the appropriate configuration file in the configuration directory.

"The HTTP Adaptor Configuration File" (page 35) explains how the file is structured and lists the properties that it defines. For instructions on creating the configuration file and configuring the HTTP adaptor to use it, see "Creating the HTTP Adaptor Configuration File" (page 37).

The HTTP Adaptor Configuration File

You can set up the HTTP adaptor to get your site's configuration by reading an HTTP adaptor configuration file (called `WOConfig.xml` by default) in the configuration directory (`/Library/WebObjects/Configuration` by default). You should have only one adaptor configuration file per web server so that it can perform load balancing effectively. (See "Load Balancing" (page 77) for details.) In addition, in a site with multiple web servers, if two servers share the configuration file, instead of deploying two sites you would be deploying the same site twice. Listing 3-1 shows a configuration file that defines a site with two application hosts (`ebruce.apple.com` and `ejays.apple.com`), each running two application instances, one of the Payroll application and the other of the HR application.

Listing 3-1 A WebObjects adaptor configuration file

```
<?xml version="1.0" encoding="ASCII"?>
<adaptor>
  <application name="Payroll" urlVersion="4">
    <instance id="1" port="2002" host="ebruce.apple.com"/>
    <instance id="2" port="2001" host="ejays.apple.com"/>
  </application>
  <application name="HR" urlVersion="4">
    <instance id="1" port="2001" host="ebruce.apple.com"/>
    <instance id="2" port="2002" host="ejays.apple.com"/>
  </application>
</adaptor>
```

The HTTP adaptor configuration file provides the HTTP adaptor with information about your site's registered application instances. The structure of the configuration file is provided in Listing 3-2 (you can also view it by opening the `woadaptor.dtd` file, located in the `/Developer/Examples/WebObjects/Source/Adaptors` directory). For information on the properties defined in the configuration file, consult Table 3-1 (page 36).

Listing 3-2 Structure of the HTTP adaptor configuration file

```
<?xml version="1.0" encoding="ASCII"?>
<!DOCTYPE WebObjectsAdaptorConfiguration SYSTEM "woadaptor.dtd">
<adaptor>
    <application name=STRING
            retries=NUMBER
            scheduler=["RANDOM"|"ROUNDROBIN"|"LOADAVERAGE"]
            dormant=NUMBER
            protocol="http"
            redir=URL
            poolSize=NUMBER
            urlVersion=["3"|"4"]
            additionalArgs="unspecified"
        >
        <instance    id=NUMBER  port=NUMBER  host=STRING
            sendTimeout=NUMBER
            recvTimeout=NUMBER
            cnctTimeout=NUMBER
            sendBufSize=NUMBER
            recvBufSize=NUMBER
            additionalArgs="unspecified"
        >
        </instance>
    </application>
</adaptor>
```

Table 3-1 The properties of the HTTP adaptor configuration file

Property	Description
`name`	This property is used by the adaptor to implement load balancing. The adaptor can load-balance only between instances with the same application name. The property can be used to create groups of instances, even when the instances share the same executable file. This argument is set automatically for instances started by wotaskd.
`retries`	The number of times a request is retried (trying several instances) if a communications failure occurs before an error page is returned to the web server.
`scheduler`	The load-balancing scheme used by the adaptor for instances of the application. The options provided by WebObjects are Round Robin, Random, and Load Average. You can also use a custom load balancer; see "Load Balancing and Adaptor Settings" (page 68) for details.
`dormant`	The number of times the adaptor skips an instance of the application before trying again.
`redir`	The URL that the user is redirected to when an instance fails to respond to a direct request.

Property	Description
poolSize	The maximum number of simultaneous connections the adaptor should keep open for each configured instance.
urlVersion	The WebObjects version to use for URL parsing and formatting. All WebObjects 4, 4.5, and 5.x applications use version 4 URLs by default.
additionalArgs	Additional information to send to the instance when it's started.
id	The instance's identification number. Must be unique for the load-balancing process to operate correctly.
port	The port on which the instance runs.
host	Specifies the network interface that an instance binds to. This argument should only be used on hosts with multiple network interfaces (IP addresses).
sendTimeout	The length of time, in seconds, that the adaptor attempts to send data to an instance of the application before giving up.
recvTimeout	The length of time, in seconds, that the adaptor waits for a response from an instance of the application before giving up.
cnctTimeout	The length of time, in seconds, before the adaptor gives up connecting to an instance.
sendBufSize	The size, in bytes, of the TCP/IP socket send buffer that's used for adaptor-to-instance communication.
recvBufSize	The size, in bytes, of the TCP/IP socket receive buffer that's used for adaptor-to-instance communication.

Creating the HTTP Adaptor Configuration File

You can define your site's configuration by writing the HTTP adaptor configuration file by hand. However, JavaMonitor provides you with an easy-to-use interface that facilitates this task.

"Deployment Tasks" (page 53) shows you how to configure your site using JavaMonitor. When you are satisfied with the configuration, you can save the settings into a configuration file by copying and pasting or by telling wotaskd to write the file.

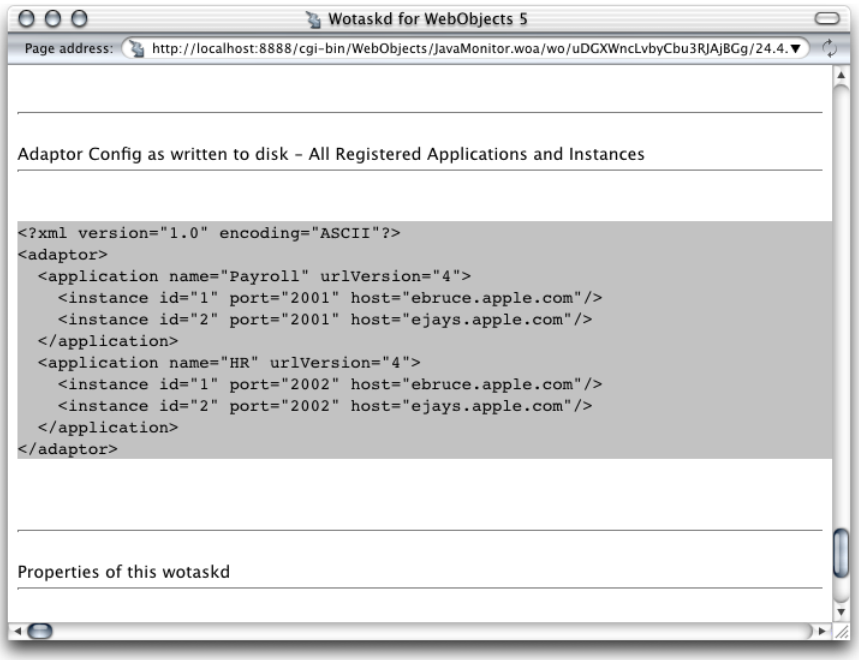
To use the copy-and-paste method, follow these steps:

1. In JavaMonitor, display the Hosts page.
2. Click YES for any host.

   The host configuration page is displayed in a new web browser window.

3. Copy the contents of the section Adaptor Config as written to disk—All Registered Applications and Instances, as shown in Figure 3-5.

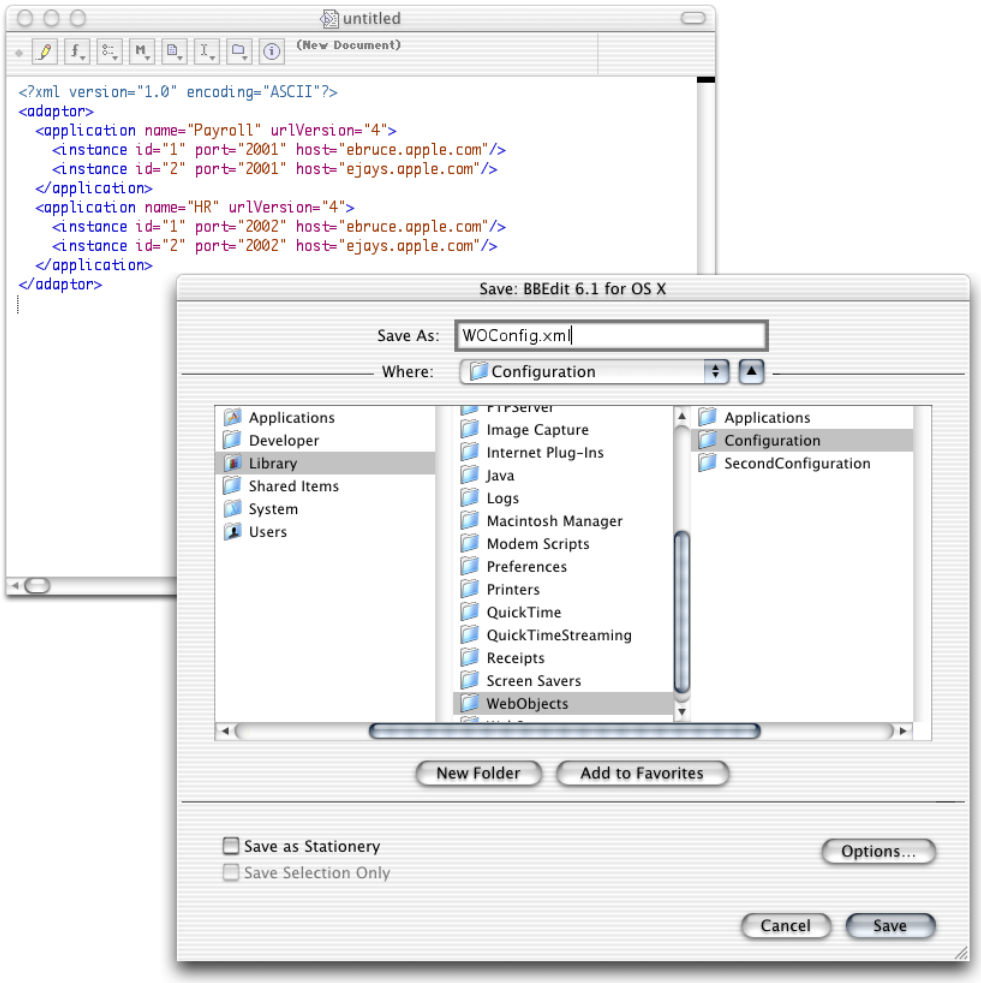
Figure 3-5 Copying the information that makes up the HTTP adaptor configuration file



4. Using a text editor, create a new file and paste the contents of the clipboard into it.

5. Save the file as `WOConfig.xml` (or any other name you choose) in the configuration directory.

Figure 3-6 Creating and saving the HTTP adaptor configuration file



If instead of copying and pasting you want wotaskd to create the configuration file for you, you can start a wotaskd process specifically to create the file or you can tell wotaskd to continually maintain the configuration file.

To start a wotaskd process specifically to create the file, you must first stop the process that corresponds to the site you configured if it's already running on the web server computer.

To start a wotaskd process that writes the configuration file to the default location, execute the following two commands using your command shell editor:

```
cd /System/Library/WebObjects/JavaApplications/wotaskd.woa
./wotaskd -WOPort <port> -WOSavesAdaptorConfiguration true
```

To specify a different location for the HTTP adaptor configuration file, see `WODeploymentConfigurationDirectory` in *WebObjects Application Properties Reference*. If you want to give the file a different name, "Setting the Name of the HTTP Adaptor Configuration File" (page 42) shows you how.

To tell wotaskd to maintain the configuration file on a permanent basis, add the following to the WOServices script line that starts the wotaskd process:

```
-WOSavesAdaptorConfiguration true
```

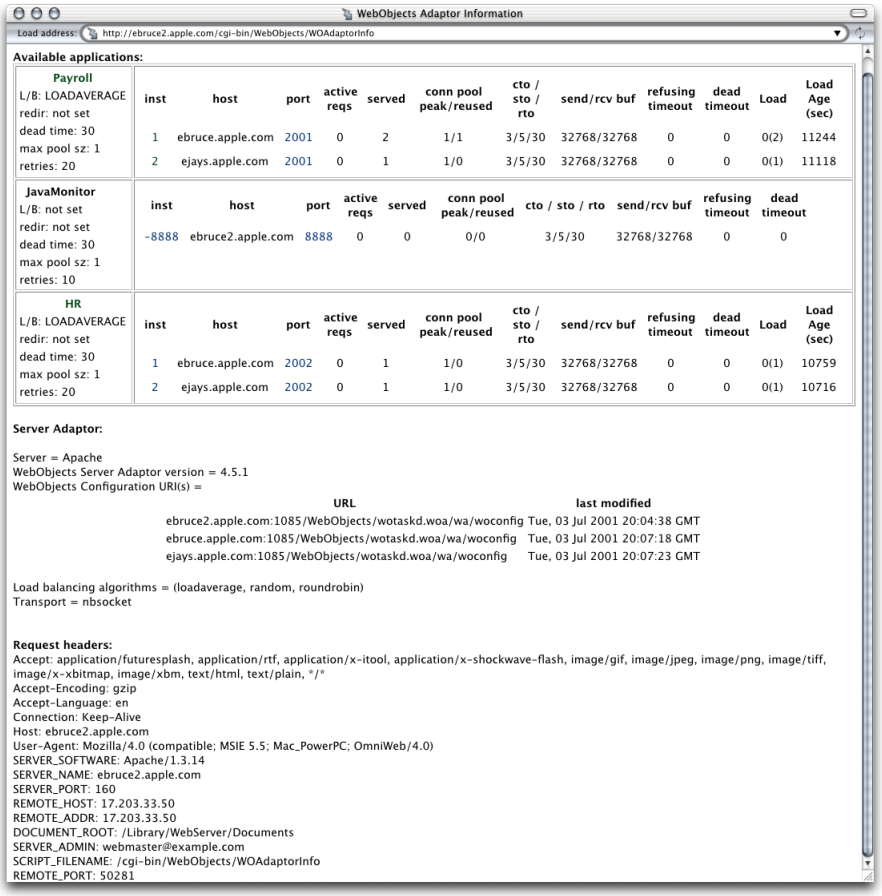
When you restart your web server, the HTTP adaptor configuration file is updated every time you make a change to your site's configuration through JavaMonitor. The changes are picked up by the HTTP adaptor the next time it reads the configuration file.

To configure the HTTP adaptor to read the configuration file instead of using a multicast request or a host list, follow the instructions in "Setting the Name of the HTTP Adaptor Configuration File" (page 42).

The WebObjects Adaptor Information Page

The WebObjects Adaptor Information page displays information about an HTTP adaptor. Access to this page is disabled by default so you must modify the adaptor configuration file to allow access. See "Setting Access to the WebObjects Adaptor Information Page" (page 43) for details. Figure 3-7 shows an example of the WebObjects Adaptor Information page.

Figure 3-7 The WebObjects Adaptor Information page



Customizing HTTP Adaptors

For the most part, you shouldn't need to modify the default values of settings in the configuration file. However, if you want to change the way the HTTP adaptor obtains your site's state information, for example, you need to perform some of the procedures explained here.

These are the tasks explained in this section:

Setting the Multicast Address and Port

The following list explains how to set the multicast address, port, and configuration refresh interval (in seconds) in the supported HTTP adaptors. The default values for each of these properties are `239.128.14.2`, `1085`, and `10` respectively. The adaptor uses the configuration interval to determine the amount of time that passes between state discoveries on your site. The host-discovery process occurs 10 times less frequently than the time indicated by the configuration refresh interval. For example, with the configuration refresh interval set to `10`, the discovery process occurs every 100 seconds.

- **Apache**: Set the value of the `WebObjectsConfig` variable in the `apache.conf` file to the desired values, using the format shown below:

  ```
  WebObjectsConfig webobjects://<address>:<port> <configuration_interval>
  ```

- **CGI**: Set the `WO_CONFIG_URL` environment variable to `webobjects://<address>:<port>`. Make sure your web server is configured to pass the variable to the adaptor (consult your web server's documentation for instructions).

Setting the Host List

The following list explains how to set a host list for a site with two hosts, `host1` and `host2`, in the supported adaptors with a configuration interval of `10` (the configuration interval cannot be set in the CGI adaptor).

- **Apache**: Set the `WebObjectsConfig` variable in the `apache.conf` file to the desired list of hosts. By default it's set to `http://localhost:1085 10` (the `10` is the configuration refresh interval). Separate each host with a comma, as shown in the following example:

  ```
  WebObjectsConfig http://host1:1085,http://host2:1085 10
  ```

- **CGI**: Set the `WO_CONFIG_URL` environment variable to `http://host1:1085,http://host:1085`. Make sure the web server is configured to pass the variable to the adaptor (consult your web server's documentation for instructions).

Setting the Name of the HTTP Adaptor Configuration File

The following list shows how to set the name and location of the HTTP adaptor configuration file for the supported HTTP adaptors.

- **Apache**: Set the value of `WebObjectsConfig` variable in the `apache.conf` file to the path of the adaptor configuration file.

  ```
  WebObjectsConfig file://<path-to-an-xml-config-file> 10
  ```

- **CGI**: Set the `WO_CONFIG_URL` environment variable to `file://<path-to-an-xml-config-file>`. Make sure the web server is configured to pass the variable to the adaptor (consult your web server's documentation for instructions).

Setting Access to the WebObjects Adaptor Information Page

You can provide access to the WebObjects adaptor information page (WOAdaptorInfo) to a specific user or to everyone. To provide access to a single user, you set the values of the `username` and `password` properties. To provide public access, set the `username` attribute to `public`. The following list explains how to provide access to the information page to a user named Joe in the supported adaptors. For the changes to take effect, you need to restart the web server.

- **Apache**: Add the following lines to the `apache.conf` file, located in the `/System/Library/WebObjects/Adaptors/Apache` directory:

  ```
  WebObjectsAdminUsername joe
  WebObjectsAdminPassword secret
  ```

- **CGI**: Set the `WO_ADAPTOR_INFO_USERNAME` and `WO_ADAPTOR_INFO_PASSWORD` environment variables to the appropriate values. Make sure the web server is configured to pass the variables to the adaptor (consult your web server's documentation for instructions).

Setting an Alias for cgi-bin in the WebObjects URL

The following list explains how to change the `cgi-bin` part of the URL used to connect to an application instance to `Store` in the Apache, and CGI adaptors:

- **Apache and CGI**: In the `apache.conf` file, change the line

  ```
  WebObjectsAlias /cgi-bin/WebObjects
  ```

  to

  ```
  WebObjectsAlias /Store/WebObjects
  ```

Setting the Document Root Path of the Web server

The following list explains how to set the path to the document root of the supported web servers.

- **Apache**: In the `apache.conf` file, change the line

  ```
  WebObjectsDocumentRoot /Library/WebServer/Documents
  ```

  to

  ```
  WebObjectsDocumentRoot <document-root-path>
  ```

- **CGI**: Set the value of the `CGI_DOCUMENT_ROOT` environment variable to the desired path. Make sure that your web server is configured to pass the variable to the adaptor (consult your web server's documentation for instructions).

Setting WebObjects Options

You can set several WebObjects options through an environment variable or Registry setting, depending on which web server you're using. Table 3-2 shows the WebObjects options available:

Table 3-2 WebObjects options

Option	Description
`logPath`	Full path to log file.
`logLevel`	Logging level used to log HTTP adaptor activity. Values (from most verbose to least verbose) `Debug`, `Info`, `Warn`, `Error`, and `User`.
`stateFile`	Filename to use for the shared state file (applicable to Apache and CGI on Unix or Mac OS X Server only). Use when running multiple HTTP adaptors on a single computer.
`redir`	The URL users are redirected to when the application or Web page they try to access is unavailable.

The following list explains how to set WebObjects options for the supported HTTP adaptors:

- **Apache**: Add the `WebObjectsOptions` variable to the `apache.conf` file:

  ```
  WebObjectsOptions redir=<redirection_url>, logLevel=Debug
  ```

- **CGI**: Set the `WEBOBJECTS_OPTIONS` environment variable to the appropriate value; for example, `redir=<redirection_url>, logLevel=Debug`. Make sure the web server is configured to pass the variable to the adaptor (consult your web server's documentation for instructions).

Managing Application Instances

This chapter provides a detailed description of JavaMonitor and wotaskd, two tools you use to manage the application instances running on your site, and it explains how instances communicate with wotaskd:

- "Configuration Files" (page 45) provides an overview of the configuration files that can be used in your site.
- "Lifebeats" (page 46) introduces lifebeats, the mechanism used by wotaskd processes to determine the status of the application instances that they manage.
- "wotaskd Processes" (page 47) explains how wotaskd processes ensure that the application instances they manage are always running. It also describes how to restrict access to wotaskd using JavaMonitor.
- "Security Issues with wotaskd" (page 48) lists the security risks when using wotaskd and JavaMonitor.
- "Starting WebObjects Services" (page 48) shows you how to configure a computer to start wotaskd and JavaMonitor during its startup process.

Configuration Files

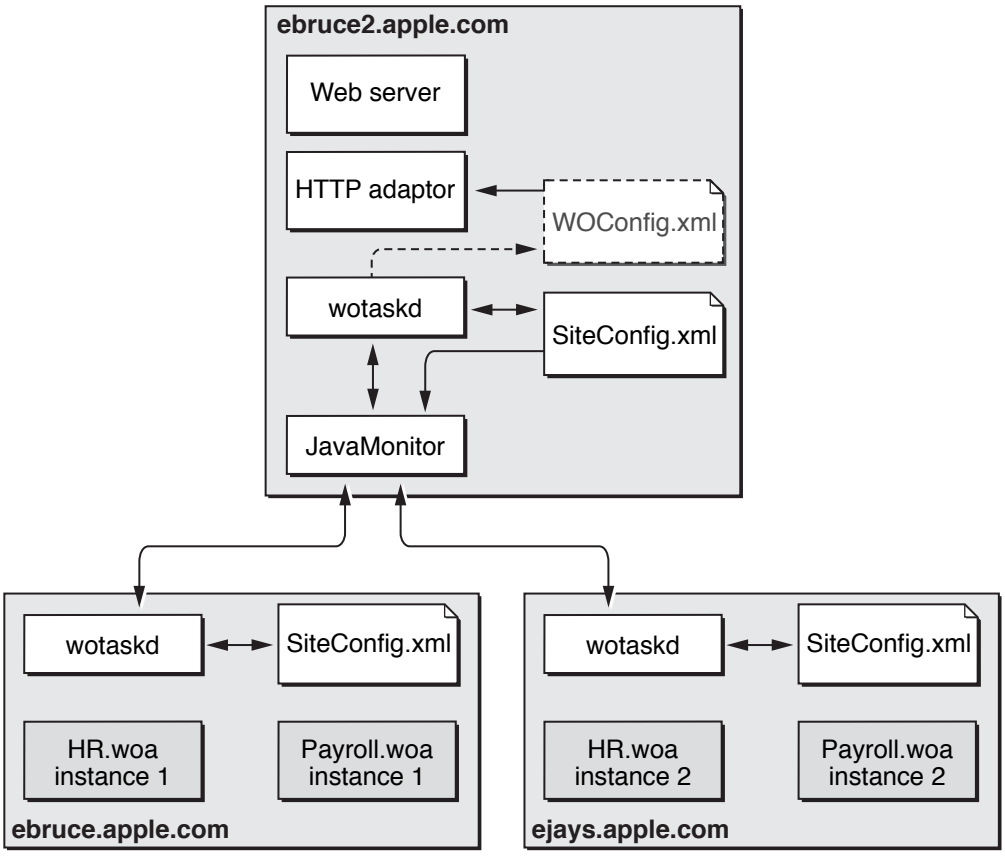
You use JavaMonitor to configure your site. With it you configure hosts, applications, and application instances. You also define schedules for restarting instances and choose the algorithm used to balance the user load among the instances of an application. For details on the tasks that you can perform using JavaMonitor, see "Deployment Tasks" (page 53).

The `SiteConfig.xml` file, which is maintained on each host's deployment-configuration directory by wotaskd, stores the configuration choices you make in JavaMonitor. You should not modify the contents of this file directly. The user under which wotaskd runs must have read and write privileges to `SiteConfig.xml` and the user under which JavaMonitor runs must have read privileges.

You can create another configuration file (the HTTP adaptor configuration file), which is called `WOConfig.xml` by default. This is the file the HTTP adaptor uses to obtain your site's configuration when you choose the configuration file method for the adaptor.

Figure 4-1 shows how the configuration files are distributed in a deployment with one web server and two application hosts. For information on how to generate the HTTP adaptor configuration file, see "Creating the HTTP Adaptor Configuration File" (page 37).

Figure 4-1 WebObjects configuration-file distribution



JavaMonitor reads the `SiteConfig.xml` file when it starts up but never writes to it. wotaskd writes the `SiteConfig.xml` file when instructed by JavaMonitor to do so—such as when existing entities are configured, added, or deleted. Furthermore, JavaMonitor tells wotaskd to update the `SiteConfig.xml` file only when you use it to change an aspect of your site's configuration.

Lifebeats

After adding an application to your site, you need to add instances of it as well. This allows the application's users connect to it. One of the main goals of WebObjects Deployment is to provide you with tools that help you deploy a resilient site. To accomplish that, wotaskd restarts application instances that become unresponsive.

A **lifebeat** is a status message that an application instance sends to a wotaskd process to keep it informed of the instance's status. There are four kinds of lifebeats:

- has started
- is alive
- will stop
- will crash

An application instance is configured by default to send lifebeats to a wotaskd process through TCP (Transmission Control Protocol) sockets. (A **socket** is a mechanism through which two processes communicate.) Instances can also send lifebeats using UDP (User Datagram Protocol) sockets. UDP sockets use fewer resources than TCP sockets.

The lifebeat mechanism is how the state of your site is constantly updated. Lifebeats are sent from a separate execution thread in a WebObjects application and do not interfere with, nor are they affected by, normal request processing.

By default, application instances try to send a lifebeat every 30 seconds. If a failure occurs (there is no wotaskd process listening on the port indicated by `WOLifebeatDesinationPort`), the instance does the following:

1. Sends up to 10 TCP lifebeats until a response is received. After the tenth unanswered lifebeat, the instance goes to stage 2.
2. Sends UDP lifebeats (low-resource–lifebeat mode) until a response is received. If the instance cannot create a UDP socket, it goes back to stage 1. When the instance receives an acknowledgement, it resumes sending TCP lifebeats.

The two-stage mechanism allows application instances to go into low-resource lifebeat mode if wotaskd isn't running when the instance starts.

wotaskd Processes

WebObjects Deployment uses wotaskd to manage the application instances running on your application hosts. Its main task is to start up instances when hosts are restarted. To accomplish this, wotaskd itself has to be restarted when the host starts up. This is done by configuring wotaskd as a service started when the computer boots. By default, a wotaskd process running on port `1085` is configured as a service on all supported platforms. The implementation of this feature is platform-specific. See "Starting WebObjects Services" (page 48) for details.

You need to run a wotaskd process on every computer that you want to use as an application host. These processes constantly receive lifebeats from the application instances they manage. Lifebeats communicate the instance's state and allow wostaskd to determine the state of the instances it oversees. A wotaskd process assumes that an application instance is dead if it does not receive a lifebeat within a certain period. For details, see `WOAssumeApplicationIsDeadMultiplier` in *WebObjects Application Properties Reference*.

When you restrict access to JavaMonitor with a password, wotaskd processes running on hosts configured in JavaMonitor are also protected: They do not respond to `http://<hostname>:<wotaskd-port>`.

To access the state information of a particular wotaskd process, you use JavaMonitor's Host page. See "Viewing a Host's Configuration" (page 56) for more information.

Security Issues with wotaskd

It is very important to secure your website using a firewall to avoid security problems. Using wotaskd and JavaMonitor with Apache but without a firewall has the following security risks:

- Passwords are sent in clear text.
- There is no authentication mechanism between wotaskd and JavaMonitor.
- The sockets used by wotaskd are not secure.

Warning: To avoid serious security problems, any computer that runs JavaMonitor and wotaskd should always be behind a firewall. In addition, only one server per subnet should run JavaMonitor at any time.

Starting WebObjects Services

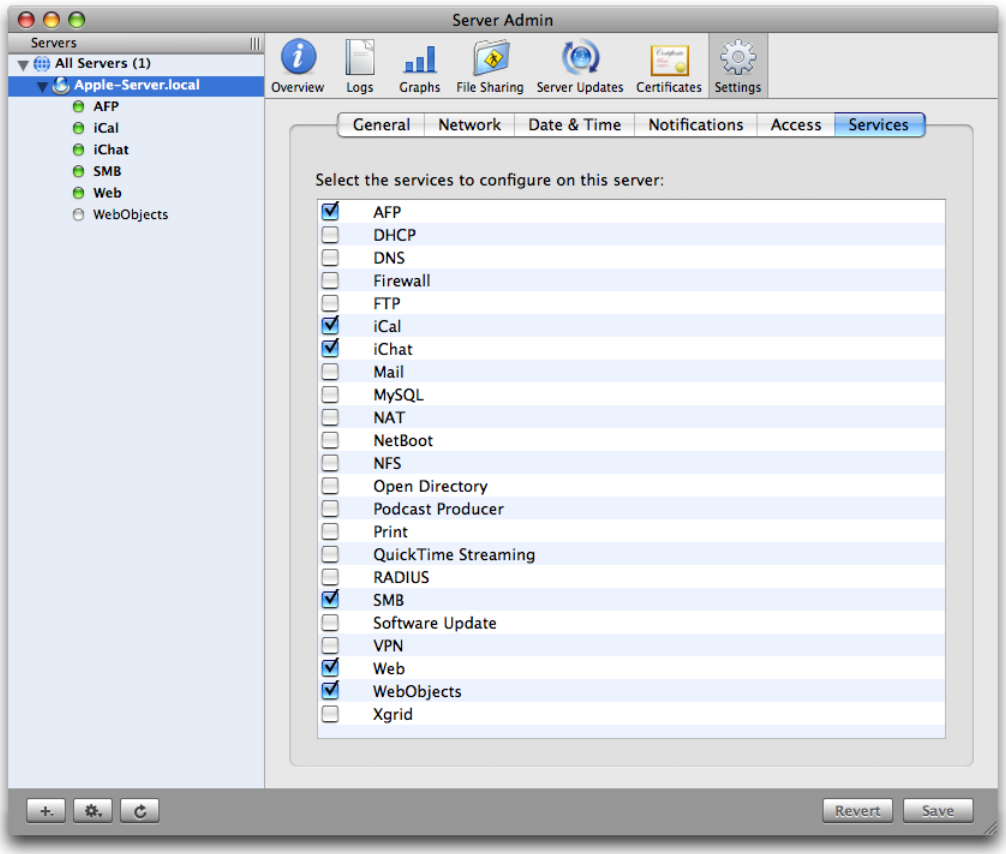
You use JavaMonitor and wotaskd to build and troubleshoot your site. WebObjects provides ways that allow you to control some of the behavior of JavaMonitor and wotaskd. Specifically, you can determine whether they are started automatically when a computer starts up. That way, you have one less item to worry about if a computer goes down. On the other hand, if you prefer a more hands-on approach to site management, you can start and stop WebObjects services manually.

Starting WebObjects Services Automatically on Mac OS X Server

To set wotaskd and JavaMonitor to start automatically upon boot, you need to first add the WebObjects module to the Server Admin application. (On Mac OS X Server v10.4 the WebObjects module is already included.) To add the WebObjects module, choose General Settings > Services to display the list of services as shown in Figure 4-2.

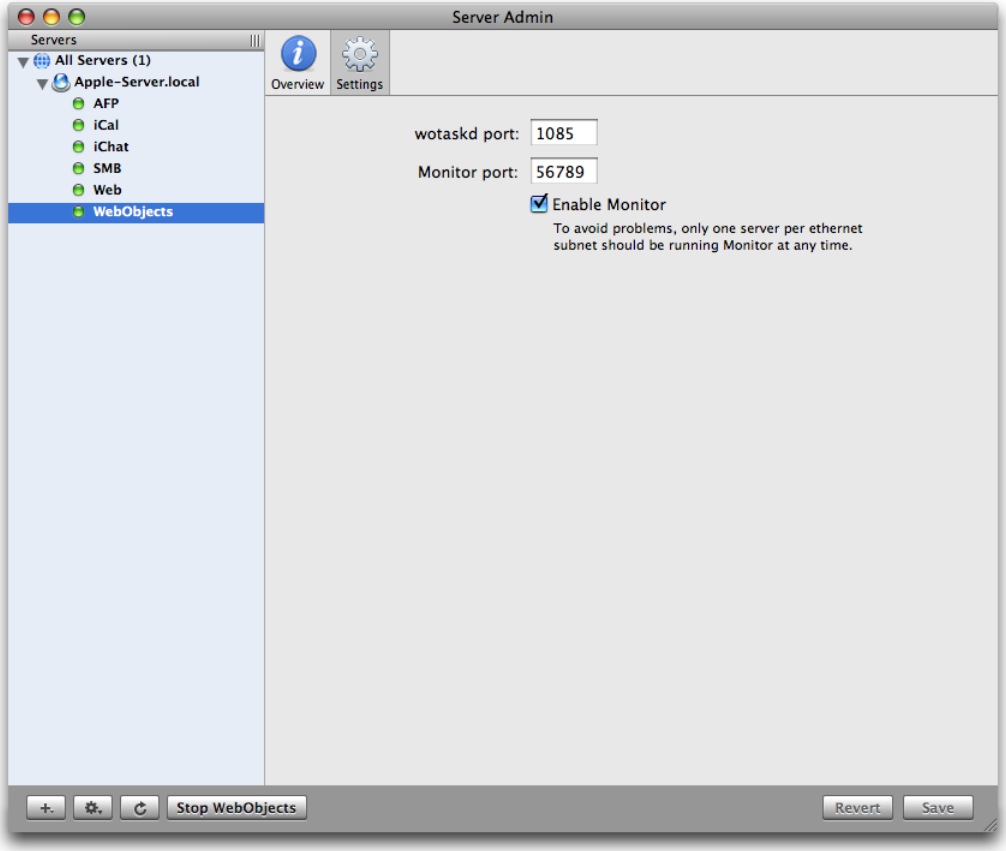
Important: On Mac OS X Server v10.5 and later, you need to select the Advanced Server Configuration instead of the Standard Configuration when doing a clean install in order to select and configure the WebObjects module later.

Figure 4-2 General Settings pane



Select the WebObjects module and save your changes to display the WebObjects pane shown in Figure 4-3. Next, select WebObjects in the outline view and select the Enable Monitor option to start JavaMonitor on boot.

Figure 4-3 WebObjects pane



You can also set wotaskd and JavaMonitor to start automatically from your command shell editor. To start wotaskd, open `/System/Library/LaunchDaemons/com.apple.wotaskd.plist` and set the `Disabled` key value to `false`. To start JavaMontior, open `/System/Library/LaunchDaemons/com.apple.womonitor.plist` and set the `Disabled` key value to `false`.

After setting these values, you need to either restart your server or use the `launchctl` application to load these property lists. For more information on `launchctl`, type `man launchctl` in your command shell editor.

Follow these steps to load the property lists manually:

1. Run launchctl as root.
2. Use the `list` command to verify that `com.webobjects.womonitor.plist` and `com.webobjects.wotaskd.plist` are listed.
3. If these files exist, load them manually using these commands:

```
load /System/Library/LaunchDaemons/com.apple.wotaskd.plist

load /System/Library/LaunchDaemons/com.apple.womonitor.plist
```

4. Start the services with the commands:

```
start com.webobjects.wotaskd

start com.webobjects.womonitor
```

5. Quit launchctl by typing `Control-c`.

Starting JavaMonitor Manually

To start JavaMonitor, enter the following commands in your command shell editor:

```
cd ($NEXT_ROOT)/System/Library/WebObjects/JavaApplications/JavaMonitor.woa
./JavaMonitor
```

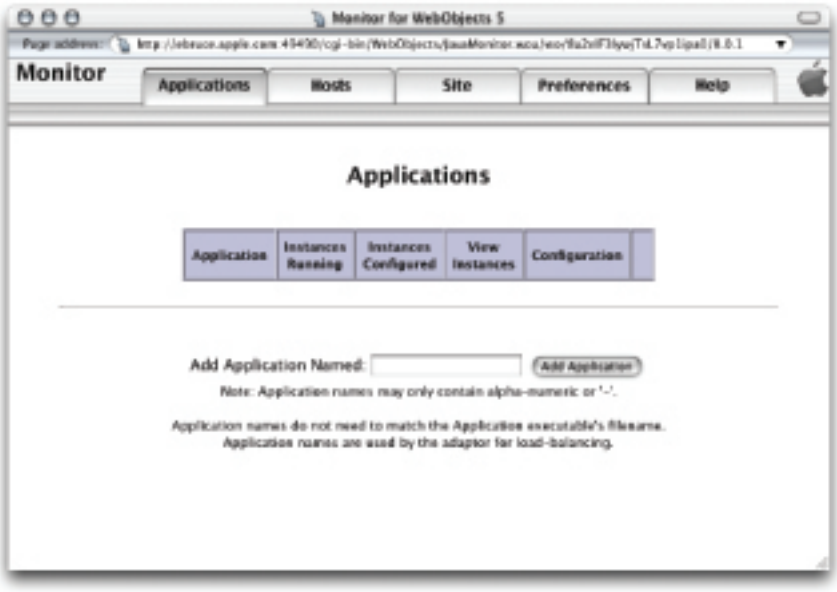
You should see output similar to that in Listing 4-1.

Listing 4-1 Starting JavaMonitor

```
Reading MacOSClassPath.txt ...
Launching JavaMonitor.woa ...
...
Creating LifebeatThread now with: JavaMonitor 49490 ebruce.apple.com/17.203.33.19
 1085 30000
Opening application's URL in browser:
http://ebruce.apple.com:49490/cgi-bin/WebObjects/JavaMonitor
Waiting for requests...
```

A page like the one in Figure 4-4 should display in your web browser. If your browser is not launched automatically, you can copy the URL from your shell and paste it into your browser's address field.

Figure 4-4 JavaMonitor—empty Applications page



Deployment Tasks

Now that you are acquainted with the deployment tools of WebObjects, you are ready to put that knowledge to work by deploying your applications.

This chapter explains how JavaMonitor allows you to perform most of the tasks required to maintain a WebObjects application site with point-and-click ease. The sections below guide you through the process of adding and configuring hosts, applications, and application instances. Also explained is the load-balancing mechanism used by the HTTP adaptor, which performs load balancing among the instances of an application (which can be distributed across more than one host) and how to set up email notifications so that you and your colleagues are notified when problems arise.

This chapter addresses the following subjects:

- "Setting Up Hosts" (page 53) explains how you add and configure hosts for application deployment. Note that this is different from configuring hosts in the HTTP adaptor.
- "Installing Applications" (page 58) shows you where to place application files and web server resources on an application host before deploying an application.
- "Setting Up Applications" (page 60) details the ways you can customize a deployment. These include choosing a load-balancing algorithm and scheduling instances to restart at regular intervals.
- "Configuring Sites" (page 74) describes the site-wide properties available. These include configuring JavaMonitor to use your SMTP (Simple Mail Transfer Protocol) server to send email notifications.
- "Setting JavaMonitor Preferences" (page 75) shows you the JavaMonitor-specific settings you can use to tailor the tool's behavior.
- "Load Balancing" (page 77) explains how load balancing distributes user load among the running instances of an application in your site.
- "Deploying Multiple Sites" (page 78) shows how you can maintain multiple sites on one set of computers.

Setting Up Hosts

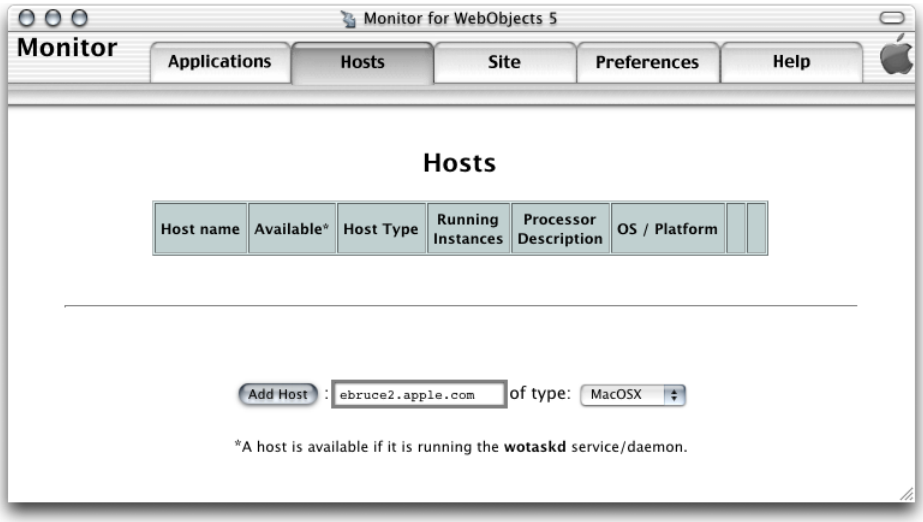
An application host is a computer that runs application instances. For JavaMonitor to be able to identify a host, the host has to be running a wotaskd process. In addition, JavaMonitor itself has to run on a registered host. That is, after launching JavaMonitor for the first time, you must add the computer it's running on as an application host, as described in "Adding a Host" (page 54).

Note: Computers running Mac OS X Server are initially set up so that they go to sleep after a period of inactivity. WebObjects does not operate reliably on a computer that's asleep. Make sure to disable idle sleep on computers on which you intend to deploy WebObjects applications.

Adding a Host

Before you can deploy applications, you need to tell JavaMonitor which hosts you want to use for deployment. (See "Load Balancing" (page 77) for additional information regarding load balancing and hosts added in JavaMonitor.) Figure 5-1 shows JavaMonitor's Hosts page, which you use to add and configure hosts.

Figure 5-1 The Hosts page



Follow these steps to add an application host to a deployment:

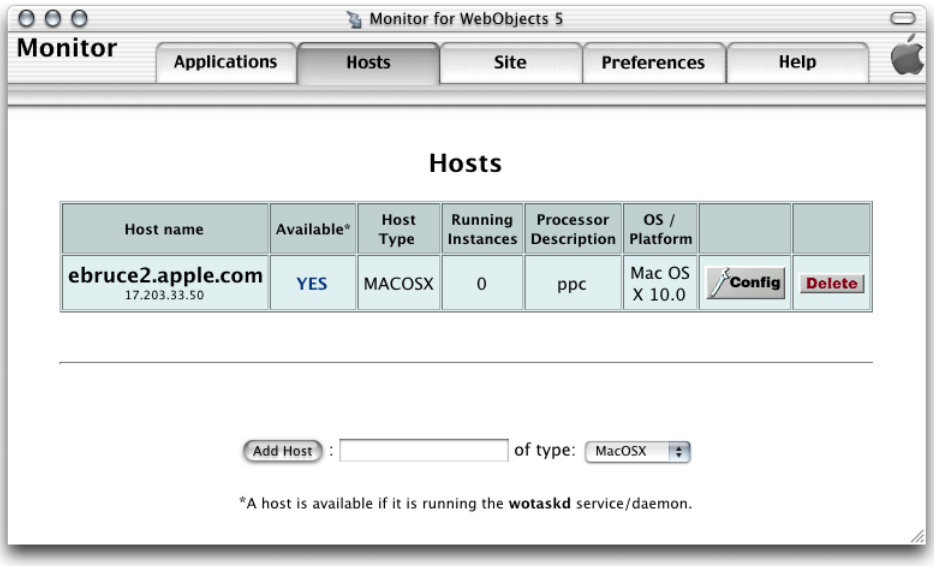
1. In JavaMonitor, click the Hosts tab.
2. Enter the name or IP address of the host you want to add in the host text input field.

   The computer must be running a wotaskd process in the port that JavaMonitor sends its lifebeats to.

   Avoid using a **loopback** address (a connection that does not go over the network), such as `localhost` or `127.0.0.1`. If you do, it must be the only application host in your site.
3. Choose the appropriate platform from the pop-up menu.
4. Click Add Host.

Figure 5-2 shows the Hosts page after a host has been added.

Figure 5-2 Newly added host in JavaMonitor



Configuring a Host

To change the configuration of an application host, click the Config button on the Hosts page. A page similar to the one in Figure 5-3 appears. It allows you to set the type of the host and to resynchronize configuration information if needed.

Figure 5-3 Host configuration page

Viewing a Host's Configuration

To display the configuration of a host, click YES, on the Hosts page. A page similar to the one in Figure 5-4 is displayed in a separate web browser window.

Figure 5-4 Host configuration information page

This page displays the current state of the host in several sections.

The section visible in Figure 5-4 shows the contents of the host's `SiteConfig.xml` file. For more information on the `SiteConfig.xml` file, see "Configuration Files" (page 45).

The next section shows the adaptor configuration sent to local HTTP adaptors, which lists all running application instances that wotaskd knows about (this includes JavaMonitor processes).

```
<?xml version="1.0" encoding="ASCII"?>
<adaptor>
    <application name="JavaMonitor">
        <instance id="-8888" port="8888" host="ebruce2.apple.com"/>
    </application>
</adaptor>
```

Note that the instance ID of the JavaMonitor process is negative. When wotaskd receives a lifebeat from an unregistered application instance (one that has not been added to your site through JavaMonitor), it discloses the instance to the web server with a negative ID number. This allows developers (internal users) to connect to development instances through the HTTP adaptor for testing purposes. To address security concerns, external users can connect to instances with negative ID numbers only if they know the instance's port number. However, this can be disabled by setting `WODirectConnectEnabled` to `false`. See `WODirectConnectEnabled` in *WebObjects Application Properties Reference* for details.

To connect to a development instance through the web server, the instance must run on the same computer that the web server runs on, and the computer must be the `localhost`.

Next is the local adaptor configuration sent to remote HTTP adaptors. This section lists all instances that are active, registered (configured through JavaMonitor), and available to external users.

```
<?xml version="1.0" encoding="ASCII"?>
<adaptor>
  <application name="Payroll" urlVersion="4">
    <instance id="1" port="2001" host="ebruce.apple.com"/>
  </application>
  <application name="HR" urlVersion="4">
    <instance id="1" port="2002" host="ebruce.apple.com"/>
  </application>
</adaptor>
```

The next section shows the contents of the HTTP adaptor configuration file. If you tell wotaskd to write the HTTP adaptor configuration file, it lists all the registered application instances on the site, whether they are running or not. (This file is identical across all the site's application hosts.) See "The HTTP Adaptor Configuration File" (page 35) for more information and an example of the file's contents.

The last section lists information on the wotaskd process's environment, including its port and multicast address.

```
The Configuration Directory is: /Library/WebObjects/Configuration/
Wotaskd is NOT writing WOConfig.xml to disk
The multicast address is: 239.128.14.2
This wotaskd is running on Port: 1085
Wotaskd is NOT responding to Multicast
WOAssumeApplicationIsDeadMultiplier is 4
The System Properties are: ...
```

Installing Applications

Before you can deploy applications on your site, you have to install them on the hosts on which you want to run instances of them. Most applications have files of two types:

- **application files,** which store the application's logic
- **Web server resources,** which store resources that can be shared among applications

You can use either Xcode or the Ant build system to install an application on a host. Using these tools, you create an application bundle, also known as the WOA bundle, containing all the files the application needs to run. However, most web applications contain web server resources, such as images. You should place these resources in the web server's `Document Root` directory. You do this that by performing a split installation.

In a split installation, a directory containing an application's web server resources is placed in the web server's `Document Root` directory. This directory mirrors the WOA bundle's organization, but it contains only web server resources. You can place the WOA bundle anywhere you like, preferably a location inaccessible to the outside world.

Building for Deployment with Xcode

To perform a split installation of your application using Xcode 2.1 or later, navigate to your project's directory and execute the following commands as the `root` user using your shell editor:

```
xcodebuild install -configuration Deployment DSTROOT=/
xcodebuild install -configuration WebServer DSTROOT=/
```

To perform a split installation of your application using Xcode 2.0 or earlier, navigate to your project's directory and execute the following commands as the `root` user using your shell editor:

```
xcodebuild install -buildstyle Deployment DSTROOT=/
xcodebuild install -buildstyle WebServer DSTROOT=/
```

Building for Deployment with Ant

Starting with WebObjects 5.4 and later, Xcode no longer supports building for deployment for new WebObjects applications based on the Ant build system. As an alternative, you can use the Java Ant build tool to do a split installation.

The steps to build for deployment described below apply to a WebObjects example project. You can find the WebObjects examples in the `/Developer/Examples/JavaWebObjects` directory.

To perform a split installation of your application using the WebObjects Ant build system, navigate to your project's directory and execute the following commands while logged in as the `root` user from a shell:

```
ant deployment
```

This command places the application files in the `/Library/WebObjects/Applications` directory and the web server resources in the `Document Root/WebObjects` directory.

The WebObjects Ant build system provides three types of application bundles in the `dist` directory of your project folder:

- The legacy WOA application bundle

  This is the traditional WebObjects WOA or Framework bundle.

- The self-contained WOA application bundle

  This self-contained WOA bundle embeds all dependent framework Jar files and resources within the bundle. The class path is configured to look for the internal bundled classes instead of the installed ones. It is generally recommended that you use a self-contained WOA bundle for deployment to keep the WebObjects application running even when the WebObjects runtime on the deployment machine is upgraded to a different version.

- A J2EE-style war bundle with an embedded WebObjects application

After producing a split installation on the development computer, copy the WOA bundle and the application's resources to their final destination (or destinations, in case you want to run instances of the application in several computers) and use JavaMonitor to configure the deployment.

You can place application files in any directory of the application host. Web server resources, however, must be placed in the `Document Root` directory of the application host. Make sure that you use the following organization:

```
<Web server Document Root>/
    WebObjects/
        AppName.woa/
            Contents/
```

```
WebServerResources/
    <resource files>
```

In most situations you should deploy an application's logic and resources on different computers. For example, while developing an application on Mac OS X, you can test it with the web server included with the system. However, you should not deploy the application in Mac OS X. Always deploy WebObjects applications on Mac OS X Server.

> **Warning:** Never remove the `WebServerResources` directory that is generated as part of the application bundle.

Setting Up Applications

To deploy an application on your site, it must be installed in the appropriate directories on the hosts that run instances of it. For more information, see "Installing Applications" (page 58).

Setting up an application in your site involves three main steps:

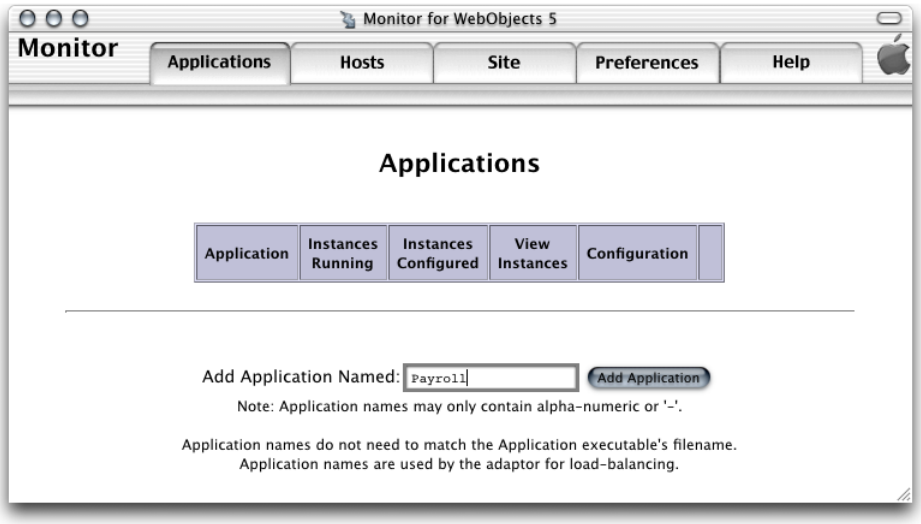
- **Adding the application,** which is described in "Adding an Application" (page 60).
- **Configuring the application**, which includes the following subtasks:
  - defining a default configuration for new instances of the application
  - defining the recipients of email notifications
  - defining a schedule (only available for existing instances)
  - choosing a load-balancing algorithm

  These steps are detailed in "Configuring an Application" (page 61).
- **Adding application instances,** which is described in "Adding Application Instances" (page 69).

Adding an Application

You add applications to your site using JavaMonitor's Applications page, shown in Figure 5-5.

Figure 5-5 Adding an application using JavaMonitor's Applications page



Follow these steps to add an application to your site:

1. Enter the application's name (without the extension) in the Add Application Named text field.
2. Click Add Application.

Configuring an Application

After you add an application, the application configuration page is displayed. This page has five major sections, which you can show and hide using the disclosure triangles:

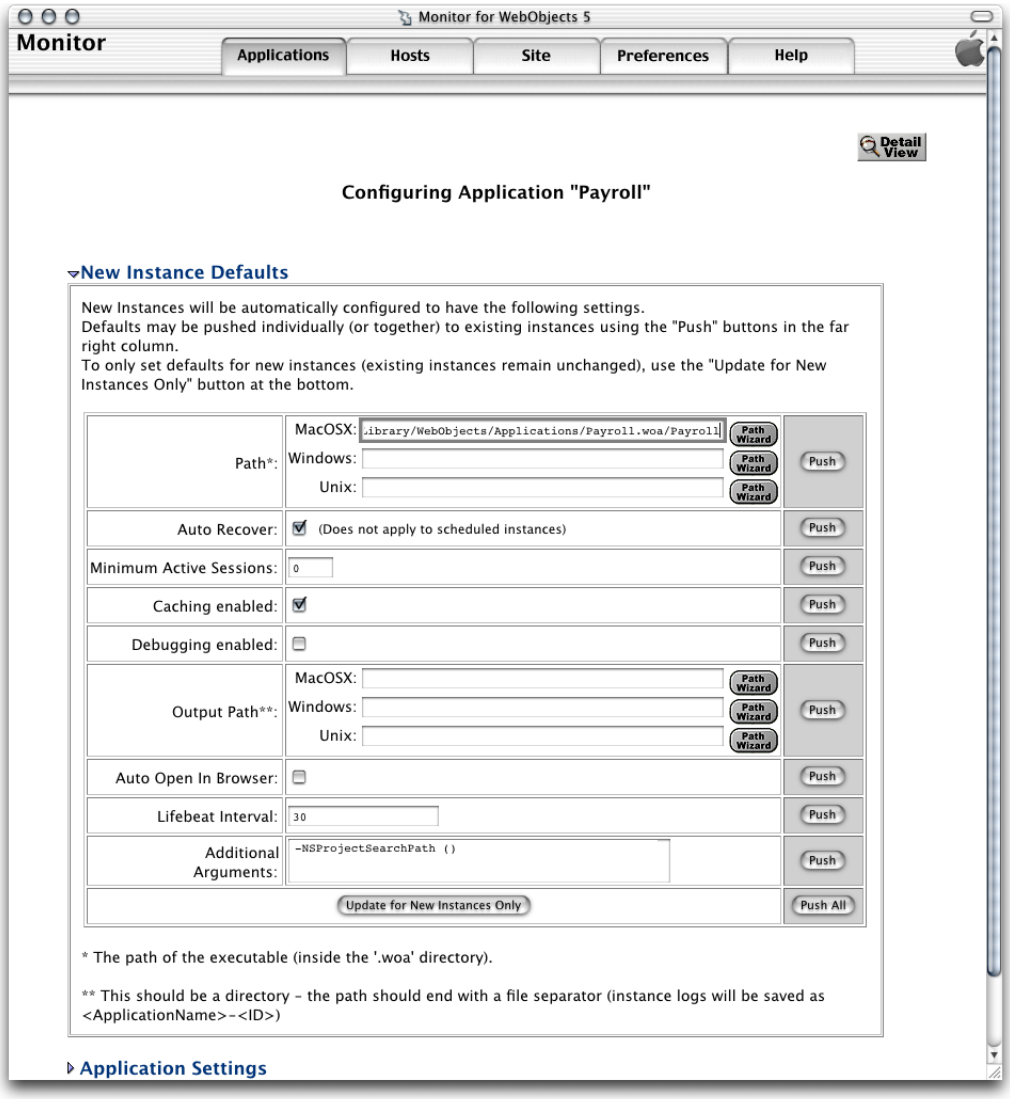
- The "New Instance Defaults" section lets you set the default values for the instance settings for application instances you add afterward. For example, you can change the default JVM setting for your application. See "New Instance Defaults" for details.
- The Application Settings section contains properties that apply to all the instances of the application. For more, see "Application Settings" (page 64).
- The Scheduling section allows you to individually schedule instances to restart at specific intervals. See "Scheduling" (page 66) for details.
- The Email Notifications section is where you specify the list of email addresses you want email notifications to be sent to. For details, see "Email Notifications" (page 67).
- The Load Balancing and Adaptor Settings section lets you choose the algorithm that the HTTP adaptor uses to perform load balancing among the instances of the application. See "Load Balancing and Adaptor Settings" (page 68).

New Instance Defaults

Figure 5-6 shows the section of the application configuration page that allows you to set defaults for the application instances. If application instances are running when you change these settings, you need to restart them to apply your changes. For details on the options shown on this page, see the description below or refer to *WebObjects Application Properties Reference* as indicated.

- Path: The file system path of the application launch script in the ".woa" directory.
- Auto Recover: Indicates whether JavaMonitor automatically restarts instances that are manually shut down or that have become unresponsive.
- Minimum Active Sessions: The minimum number of active sessions an instance must have before JavaMonitor can terminate it.
- Caching Enabled: See the description of `WOCachingEnabled` in *WebObjects Application Properties Reference*.
- Debugging Enabled: See the description of `WODebuggingEnabled` in *WebObjects Application Properties Reference*.
- Output Path: See the description of `WOOutputPath` in *WebObjects Application Properties Reference*.
- Auto-Open in Browser: See the description for `WOAutoOpenInBrowser` in *WebObjects Application Properties Reference*
- Lifebeat Interval: See the description of `WOLifebeatInterval` in *WebObjects Application Properties Reference*.

Figure 5-6 The New Instance Defaults section of the application configuration page



Here's an explanation of the buttons on the page:

- **Push**: updates the value of the property for new and configured (registered) instances of the application. The changes take effect after the instances are restarted.
- **Push All**: updates the value of the all the properties for new and configured instances of the application. As with Push, the changes become effective when the instances are restarted.
- **Update for New Instances Only**: sets the defaults to be used when you create new instances of the application. The properties of existing instances are not changed.
- **Path Wizard**: opens a tool that allows you to navigate through a host's file system.

Changing the JVM Size

By default, the maximum allowed RAM usage for a Java application is 64 MB. Changing this setting using JavaMonitor affects only the applications launched by JavaMonitor. Follow these steps to change the JVM size used by applications deployed with JavaMonitor:

1. Double-click the Applications tab in JavaMonitor.
2. In the Applications view, click the Config icon to bring up the Configuration window.
3. Type `-Xmx128m` in the Additional Arguments text field.

The `-Xmx128m` option sets the maximum RAM usage to 128 MB, but that is only an example. Experiment with different values to find out what has the best performance for your application. Setting the value too high may cause temporary slowdowns during garbage collection or disk paging. Some applications may benefit from setting the minimum RAM usage equal to the maximum. You can do this by setting the Java VM options to `-Xms128m -Xmx128m`

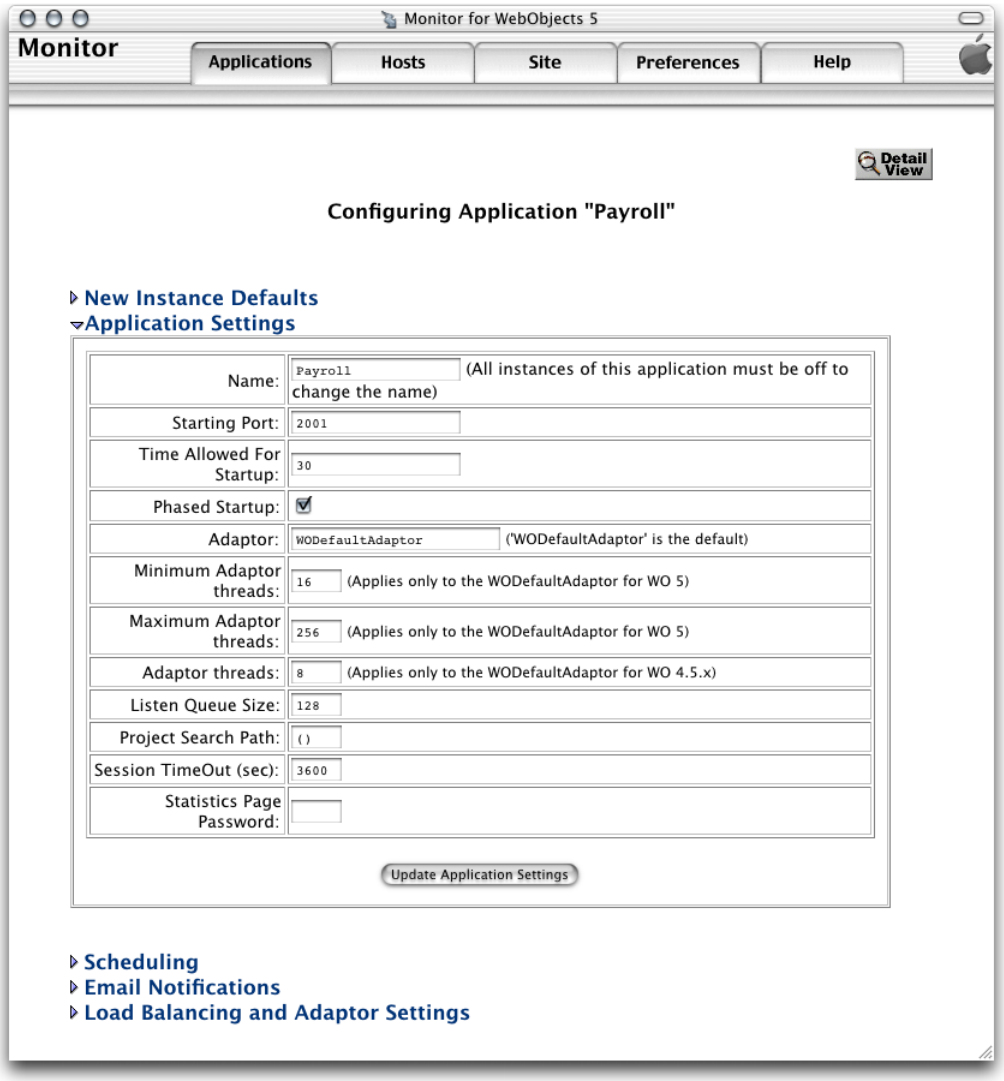
Application Settings

Figure 5-7 (page 66) shows the Application Settings section of the application configuration page. In it, you define application settings that apply to all the instances of the application. For details of the properties shown in this section, see the description below or refer to *WebObjects Application Properties Reference* as indicated.

- Name: This property is used by the HTTP adaptor to implement load balancing. The adaptor can load-balance only between instances with the same application name. The property can be used to create groups of instances, even when the instances share the same executable file. See the description of the `WOApplicationName` application property in *WebObjects Application Properties Reference*.
- Starting Port: The first port number JavaMonitor tries to assign to a new instance of the application. If the port is in use by another instance of any application, JavaMonitor uses the next unassigned port number.
- Time Allowed for Startup: The number of seconds JavaMonitor waits for an instance to start before determining that the instance failed to start.
- Phased Startup: If this option is selected, when a wotaskd process starts up, the instances that it manages that have Auto Recover selected are started one at a time instead of all at once. This option is selected by default. For more information, see the description of Auto Recover in "New Instance Defaults" (page 62).
- Adaptor: The default adaptor class an instance uses (This is not the HTTP adaptor used by the web server.) You use this if the application defines a subclass of the WOAdaptor class to be used to create adaptor objects (instead of using the WOAdaptor class itself).
- Minimum Adaptor Threads: The initial number of worker threads the adaptor creates to handle requests to the application. It applies only to adaptor processes of the WODefaultAdaptor class in WebObjects 5.
- Maximum Adaptor Threads: The maximum number of worker threads the HTTP adaptor creates to handle requests to the application. It applies only to adaptor processes of the WODefaultAdaptor class in WebObjects 5. The purpose of these threads is to process TCP or UDP packets; they have nothing to do with request processing.

- Adaptor Threads: The number of worker threads that the HTTP adaptor creates to handle requests to the application. It applies only to adaptor objects of the WODefaultAdaptor class in WebObjects 4.5.
- Listen Queue Size: Determines the depth of the listen queue. While the instance handles a request, the socket buffer can hold as many additional requests as this setting indicates before it starts refusing them. Keep in mind that if an application instance's listen queue size becomes full and a request is refused by it, the request does not get redirected to another instance with space left over in its queue. The client has to resend the request. If you expect spikes in the traffic level of a specific application, consider increasing the listen queue size. Doing so does not necessarily improve performance or allow the instance to process more requests at sustained high loads, but it may reduce the number of times an application's user must resend a particular request during high-traffic periods.
- Project Search Path: List of file-system paths that WebObjects searches in rapid-turnaround mode. These are the paths where your projects are located (relative paths are relative to your project's launch script). WebObjects searches for your projects in the order in which you specify the paths. You must specify the paths using the following format: `("firstPath","secondPath",...)`
- Session TimeOut: Specifies the time interval, in seconds, since the last request is processed after which the session times out and is terminated.
- Statistics Page Password: Specifies the password that must be entered to gain access to the instance statistics page of an application instance.

Figure 5-7 The Application Settings section of the application configuration page



Scheduling

Figure 5-8 shows the Scheduling section of the application configuration page. After you add instances of an application, you can schedule them individually here. For details, see the description below or *WebObjects Application Properties Reference*.

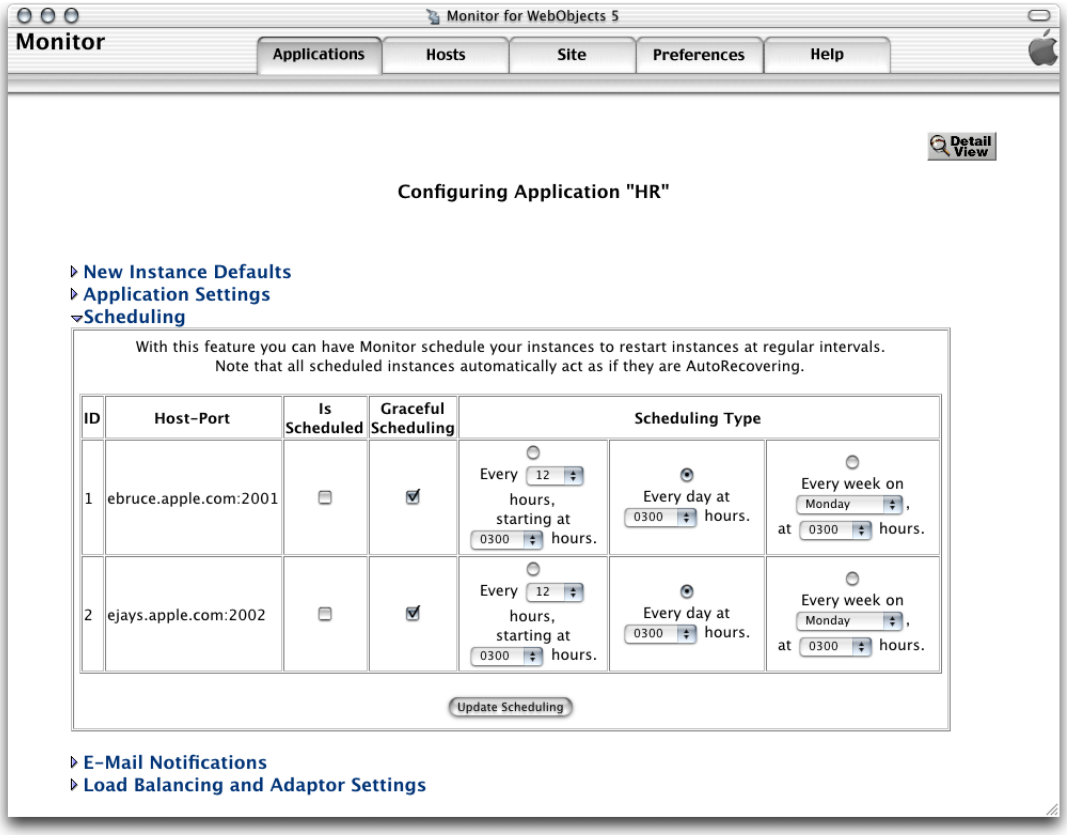
- Graceful Scheduling: Determines whether an instance is shut down gracefully or immediately at its scheduled shut-down time. During a graceful shutdown, the instance does not create sessions for new users (they are automatically directed to other available instances, if any). Existing sessions remain active until they time out or are destroyed programmatically. When the number of active sessions drops to the value set for the Minimum Active Sessions property, the instance is restarted. When Graceful Scheduling is not selected for the instance, it is restarted immediately, terminating active sessions. See description of Minimum Active Sessions in "New Instance Defaults" (page 62) for more information.

- Is Scheduled: Determines whether the schedule defined for a particular instance is active.
- Types of Schedule:

  There are three types of schedule available for instances:

  - Hourly: The instance is restarted after a certain number of hours from a particular hour.
  - Daily: The instance is restarted at a particular hour every day.
  - Weekly: The instance is restarted a particular hour on a specific day of the week.

Figure 5-8 The Scheduling section of the application configuration page

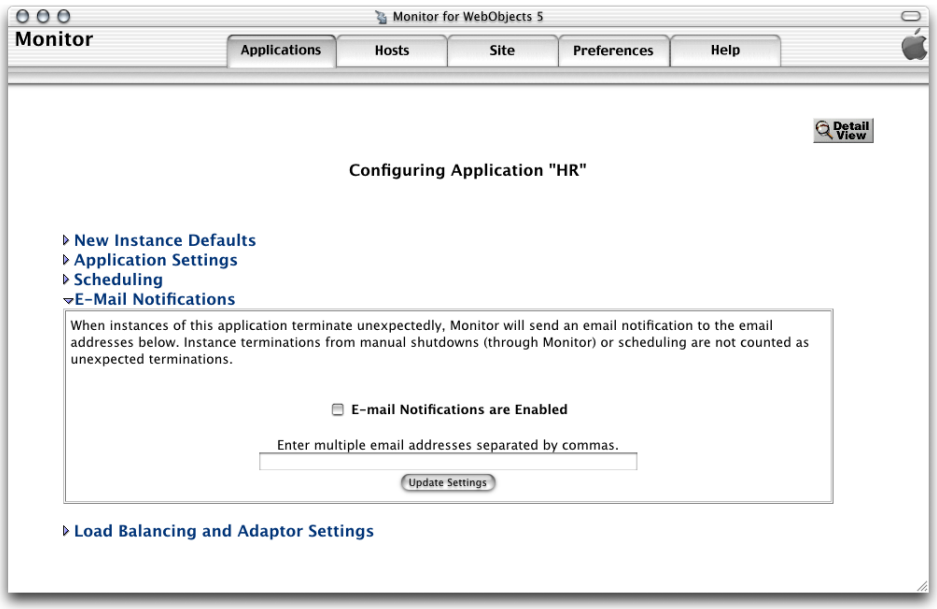


Email Notifications

Note: Before you can configure email notifications, you have to tell JavaMonitor which SMTP server to use. See "Configuring Sites" (page 74).

Figure 5-9 shows the Email Notification Settings section of the application configuration page. In it you can enter the email addresses of people that are to be notified when instances of the application terminate unexpectedly. For more information, see *WebObjects Application Properties Reference*.

Figure 5-9 The Email Notifications section of the application configuration page



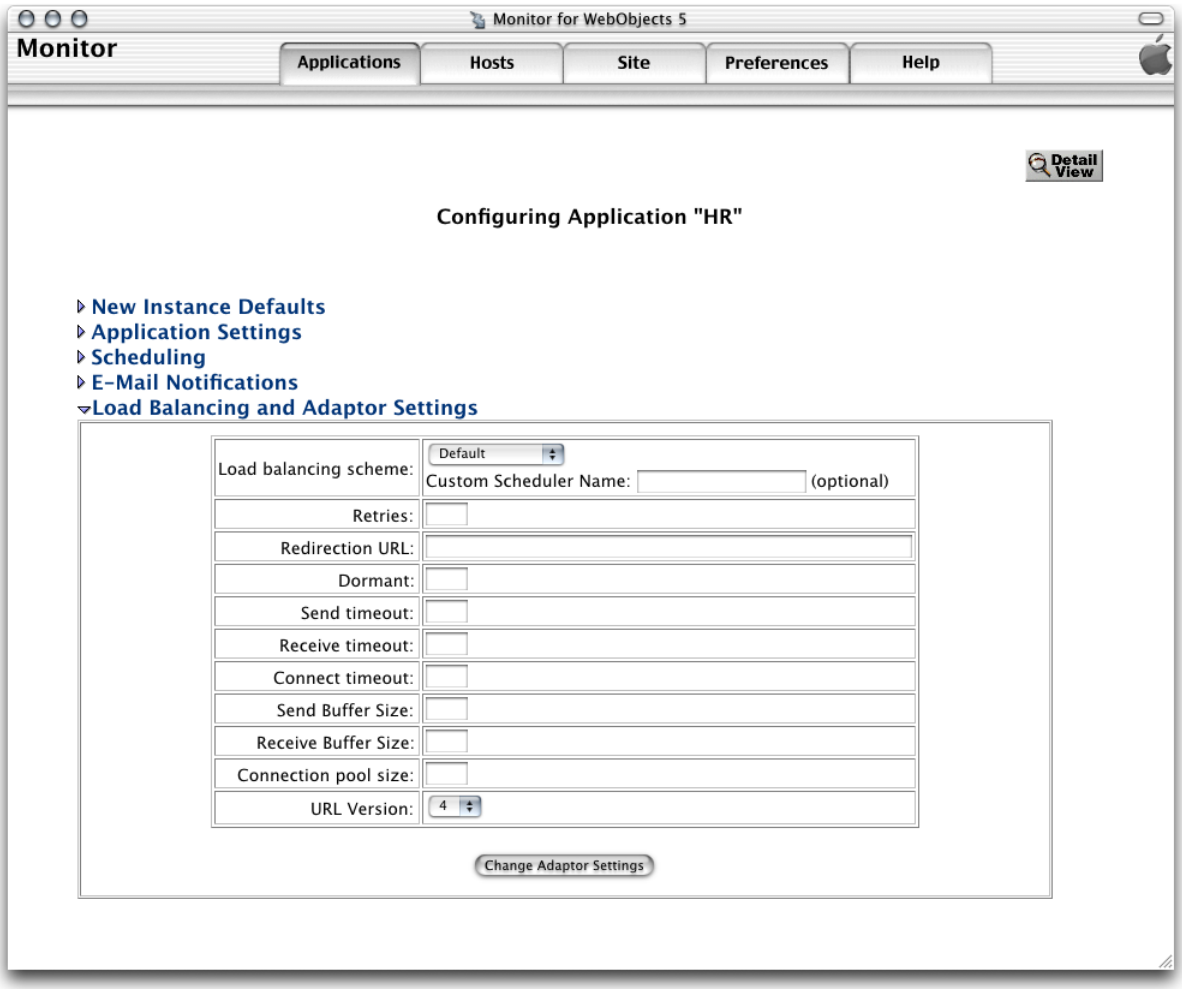
Load Balancing and Adaptor Settings

Figure 5-10 shows the Load Balancing and Adaptor Settings section of the application configuration page. This is where you enter values for the HTTP adaptor's configuration properties, including the load-balancing algorithm the adaptor uses to balance user load among the application instances. These settings override the values entered in the HTTP Adaptor Settings section of the Site page. For information on the properties you can set, see the description below or *WebObjects Application Properties Reference* as indicated.

- Load-Balancing Scheme: The load-balancing method used by the adaptor for instances of the application. The options available are Round Robin, Random, and Load Average. You can also use a custom load balancer by choosing Custom in the pop-up menu and entering the load balancer's name in the Custom Scheduler Name text field.
- Retries: The number of times a request is retried (trying several instances) if a communications failure occurs before an error page is returned to the web server.
- Redirection URL: The URL to which the user is redirected when an instance fails to respond to a direct request.
- Dormant: The number of times the adaptor skips an instance of the application before trying again.
- Send Timeout: The maximum number of seconds before the adaptor gives up attempting to send data to an instance of the application.
- Receive Timeout: The length of time, in seconds, the adaptor waits for a response from an instance of the application before giving up.
- Connect Timeout: The maximum number of seconds before the adaptor gives up connecting to an instance.
- Send Buffer Size: The size, in bytes, of the TCP socket send buffer used for adaptor-to-instance communication.

- Receive Buffer Size: The size, in bytes, of the TCP socket receive buffer used for HTTP adaptor-to-instance communication.
- Connection Pool Size: The maximum number of simultaneous connections the HTTP adaptor should keep open for each configured instance.
- URL Version: The WebObjects version to use for URL parsing and formatting. All WebObjects 4, 4.5, and 5 applications use version 4 URLs by default.

Figure 5-10 The Load Balancing and Adaptor Settings section of the application configuration page

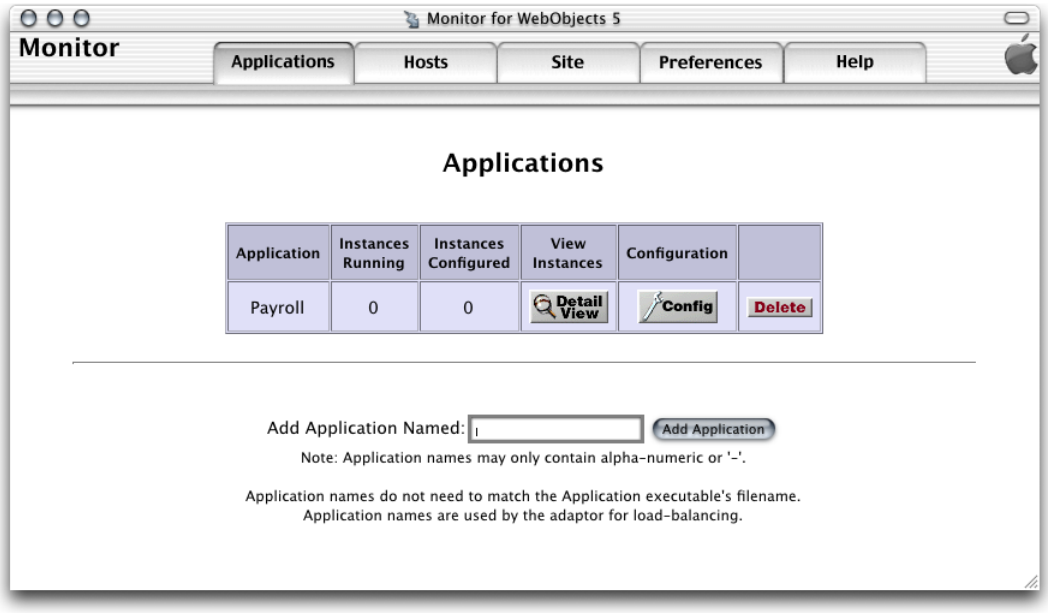


Adding Application Instances

After configuring an application in JavaMonitor, you can create application instances with ease.

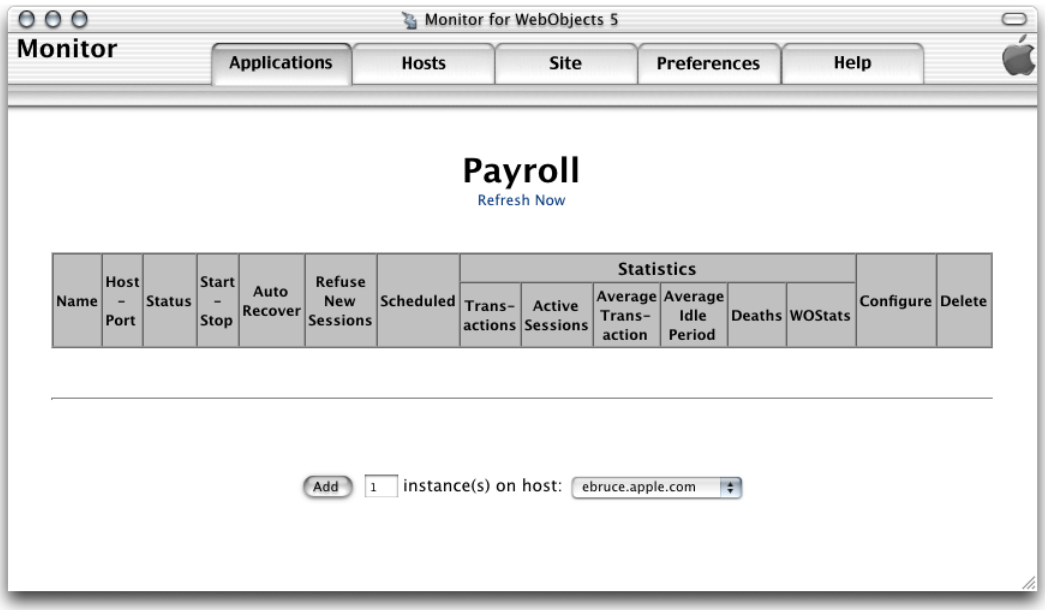
1. In JavaMonitor, click the Applications tab. The Applications page is displayed, as shown in Figure 5-11.

Figure 5-11 The Applications page with one application



2. Click the Detail View button under View Instances. The application detail page is displayed, as shown in Figure 5-12.

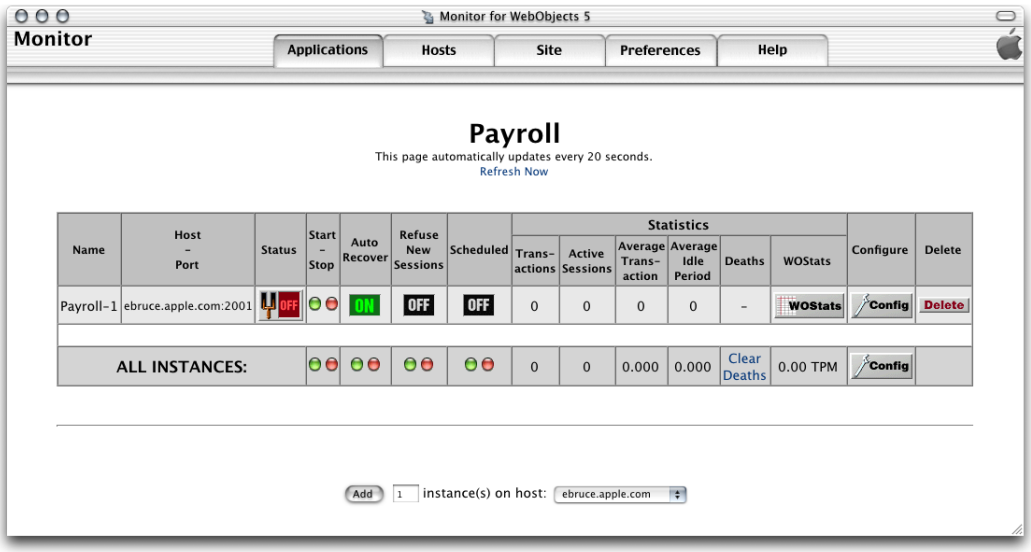
Figure 5-12 The application detail page



3. Enter the number of instances you want to add in the text input field.
4. Choose the application host you want the instances to run on from the pop-up menu.

   The application must be installed on the host you choose; otherwise, an error message is displayed when you try to start the instance. See "Installing Applications" (page 58) for details.
5. Click Add. Your web browser displays a page like the one in Figure 5-13.

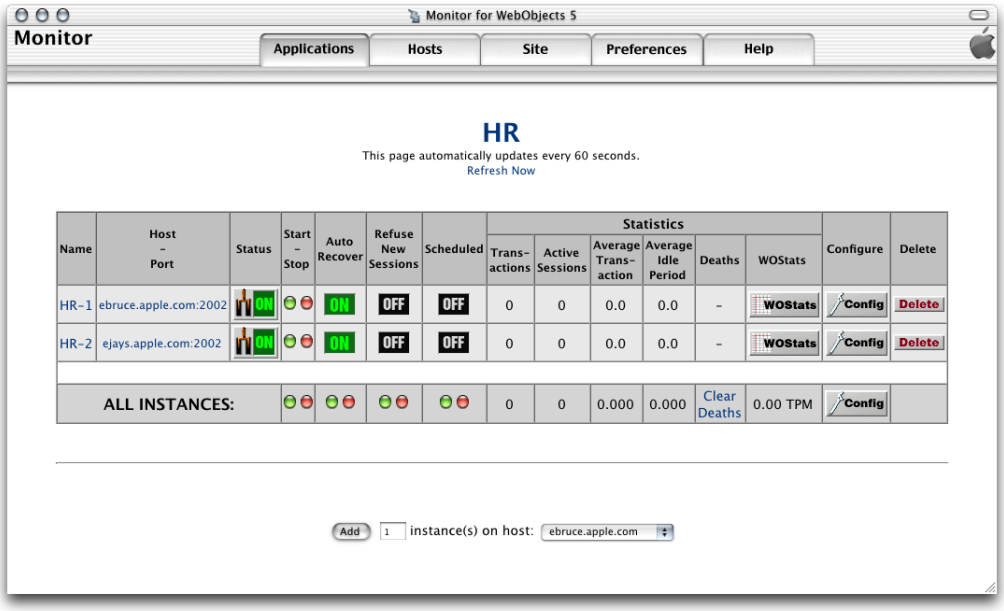
Figure 5-13 The application detail page after an instance has been added



The Status column indicates whether the instance is on or off. The first time the page is displayed, the newly added instances are off. After a moment (or if you click Refresh Now), and if Auto Recover is enabled for the instance, the page refreshes, showing that the instance is active. For details on Auto Recover, see the description of Auto Recover in "New Instance Defaults." (Read "Setting JavaMonitor Preferences" (page 75) for how to change the length of the interval between the automatic updates of the application detail page.)

Figure 5-14 shows the application detail page of the HR application, with two instances configured.

Figure 5-14 The application detail page with two instances added



The following list describes the instance configuration information that appears in the application detail page:

- **Page heading**: A link to the application through the HTTP adaptor. When you click it, the adaptor uses load balancing to determine which of the application's instances should receive the request. Then, your web browser displays a new window showing your application's entry page. For this to work, the HTTP adaptor URL has to be set ("Configuring Sites" (page 74) shows you how to do this).
- **Name**: A link to the instance through the HTTP adaptor. When you click it, the request goes through the adaptor but it's not load balanced. The URL is derived in the same way the URL of the page heading is derived but with the addition of the instance number. Such a URL looks similar to the following:

  ```
  http://ebruce2.apple.com/cgi-bin/WebObjects/HR.woa/1
  ```

- **Host-Port**: A direct link to the instance; does not go through the HTTP adaptor.
- **Status**: Tells whether the instance is on, off, starting, or stopping.
- **Start-Stop**: Click the green button to turn the instance on or the red button to turn it off.
- **Auto Recover**: Click to toggle between ON and OFF. This is available only if the instance is not scheduled. See description of Auto Recover in "New Instance Defaults" (page 62).
- **Refuse New Sessions**: Click to toggle between ON and OFF. When ON, the instance does not accept new users. This is available only if the instance is not scheduled.
- **Scheduled**: Click to toggle between ON and OFF. When ON, JavaMonitor uses the schedule defined for the instance.
- **Configure**: Click the Config button to go to the instance configuration page for the instance.
- **Delete**: Click the Delete button to delete the instance. JavaMonitor displays a confirmation page before executing the deletion.

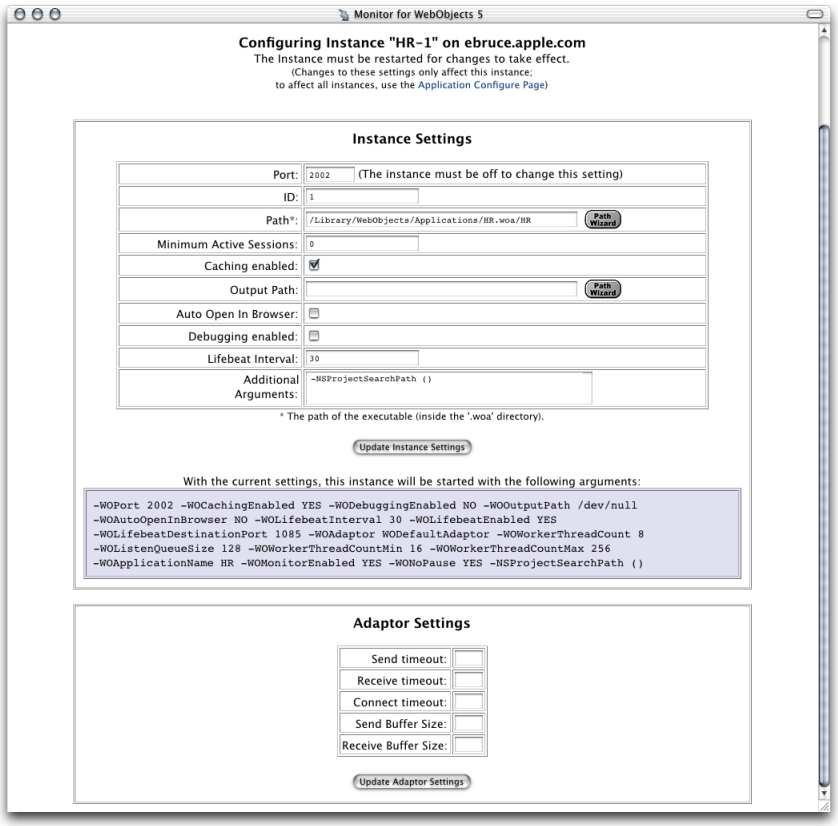
For an explanation of the columns under Statistics, see "The Application Detail Page" (page 82).

The row with the caption ALL INSTANCES contains buttons that perform some of the functions listed above on all the instances of the application. Clicking Config displays the application configuration page.

Configuring Instances

After adding an instance, you can change its configuration in the instance configuration page, shown in Figure 5-15. You can access this page through the instance's Config button in the application detail page. It contains two sections: Instance Settings and Adaptor Settings. For information on those sections, see "Instance Settings" (page 73) and "Adaptor Settings" (page 74).

Figure 5-15 Instance configuration page



Instance Settings

This section is very similar to the New Instance Defaults section of the application configuration page ("New Instance Defaults" (page 62)). It has two additional properties: ID and Port, which can be changed only after an instance has been added. ID is the application instance number which must be unique for each instance of an application. For Port details, see `WOPort` in *WebObjects Application Properties Reference*.

Adaptor Settings

In this section you can change a subset of the properties available in the Load Balancing and Adaptor Settings section of the application configuration page. For details, see the descriptions in "Load Balancing and Adaptor Settings" (page 68).

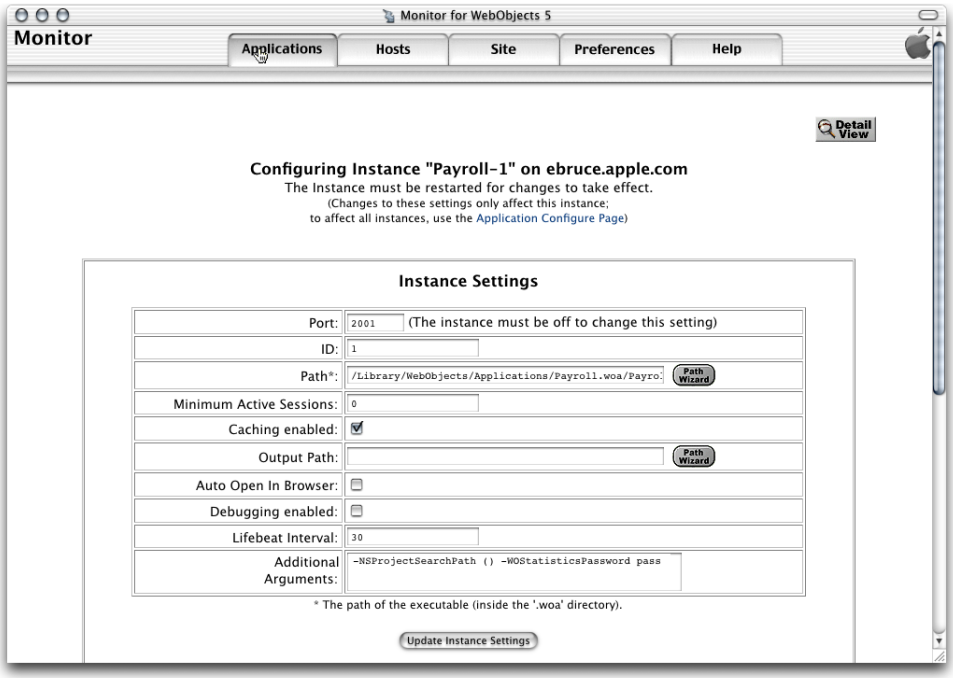
Setting a Password for the Instance Statistics Page

For each instance of your application, there's a statistics page that displays information such as its running time and memory usage. See "The Instance Statistics Page" (page 84) for more information on this page. If you want to prevent outside agents from gaining access to the instance statistics page, you can set a password in the Instance Settings section of the instance configuration page. Add the following to the Additional Arguments property:

```
-WOStatisticsPassword password
```

Figure 5-16 shows an example where the `WOStatisticsPassword` argument has been added to the Additional Arguments field.

Figure 5-16 Setting a password for an instance's statistics page



Configuring Sites

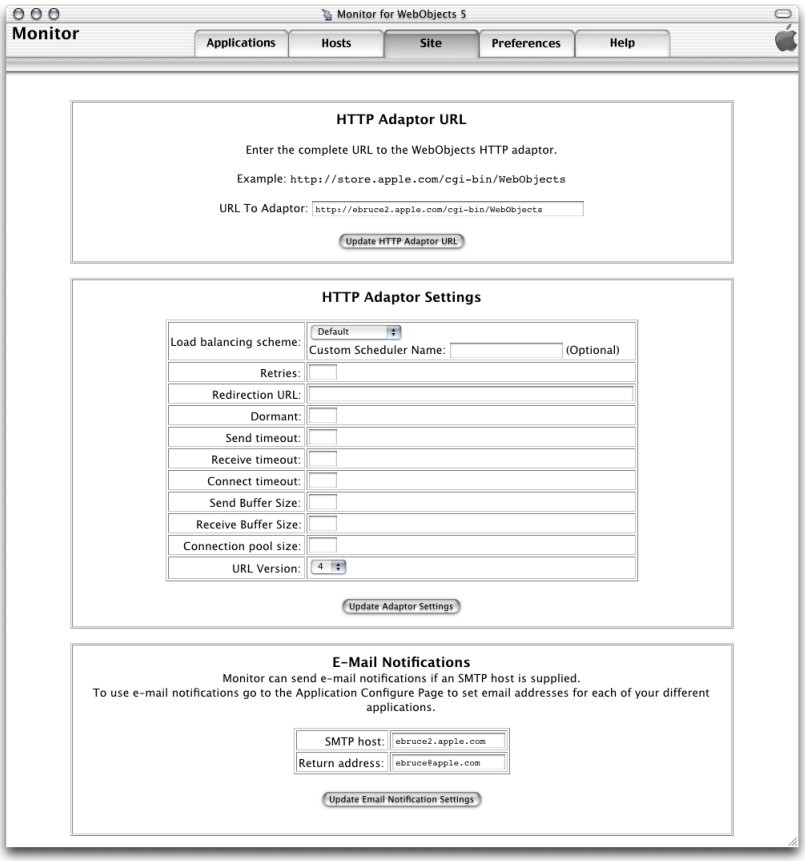
When you click the Site tab in JavaMonitor, the site configuration page is displayed. It contains three sections:

- **HTTP Adaptor URL**. This is where you tell JavaMonitor how to compose an application's URL, which is used in the application detail page to connect you to instances of your application.

  To set the URL for your application, enter the URL in the URL to Adaptor field.
- **HTTP Adaptor Settings**. This is where you set default HTTP adaptor settings for all your deployed applications. They can be overridden by each application. For more information, see the descriptions in "Load Balancing and Adaptor Settings" (page 68).
- **Email Notifications**. Here's where you specify the SMTP host and the return address that JavaMonitor uses for email notifications.

Figure 5-17 shows the site configuration page.

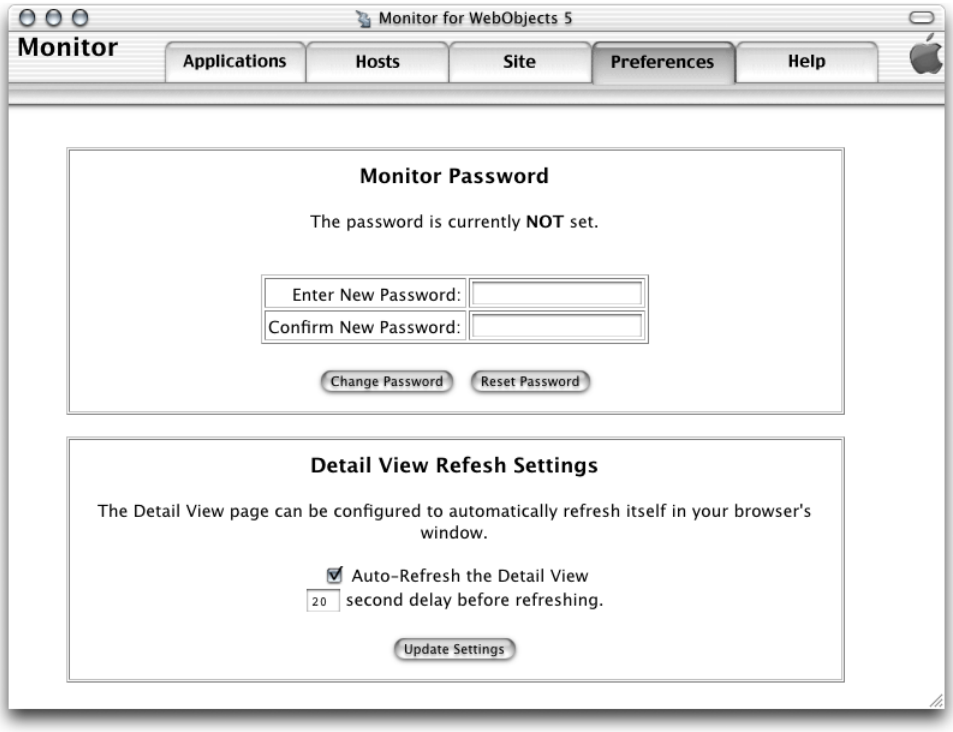
Figure 5-17 The site configuration page



Setting JavaMonitor Preferences

When you click the Preferences tab in JavaMonitor, the page in Figure 5-18 is displayed. It contains two sections: JavaMonitor Password and Detail View Refresh Settings.

Figure 5-18 The preferences page

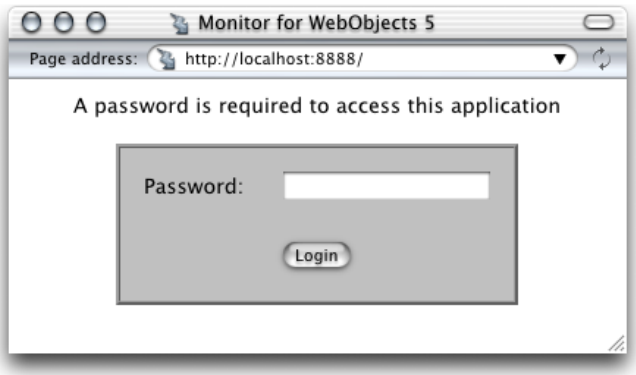


JavaMonitor Password

You can restrict access to JavaMonitor by requiring its users to enter a password before they can use it. When a site is protected this way, the site's wotaskd processes are also protected; that is, you cannot directly obtain a wotaskd process's information by connecting to its port.

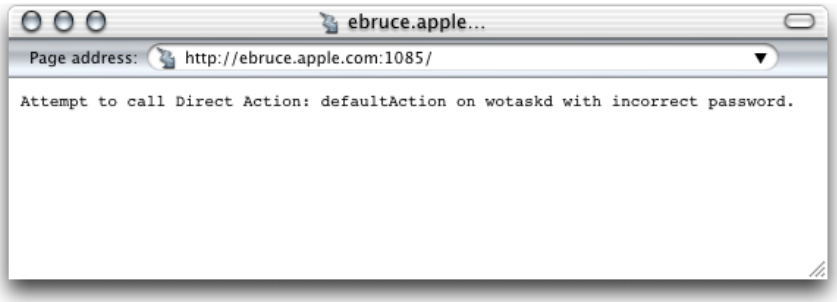
Figure 5-19 shows the login page that JavaMonitor displays after you password-protect your site.

Figure 5-19 Login page displayed by JavaMonitor on a password-protected site



When you try to view the configuration of an application host on a site that you've password-protected by connecting to the appropriate wotaskd process's port, you see a page similar to the one shown in Figure 5-20 (the page's content varies according to your deployment platform).

Figure 5-20 Page returned by wotaskd when the site is password-protected



On password-protected sites, you have to use JavaMonitor to view an application host's configuration.

Detail-View Refresh Settings

This section allows you to tell JavaMonitor if you want it to refresh the application detail page and how often to do it.

Load Balancing

Note: Load balancing occurs only between the hosts that the HTTP adaptor knows about. Adding a host in JavaMonitor is not enough. For more information on how to configure hosts in the HTTP adaptor, see "State Discovery" (page 32).

Load balancing is a mechanism by which user-load is spread out among the instances of an application; these instances can be running on different hosts. Load balancing ensures that your site's hardware resources are used efficiently and with the highest level of performance. The default load-balancing scheme used is Random. The following list describes how user load is distributed under each of the provided algorithms:

- **Random**: Assigns a user to an arbitrarily-chosen instance.
- **Round Robin**: Assigns users among instances sequentially.
- **Load Average**: Balances load by distributing users evenly among instances.

You can choose a load-balancing algorithm at two levels:

- **Site level**: You set a site-wide load-balancing algorithm in the HTTP Adaptor Settings section of the Site page. From then on, the applications in your site will use that load-balancing algorithm.

- **Application level**: You can override the site-wide load-balancing algorithm in the Load Balancing and Adaptor Settings section of the application configuration page.

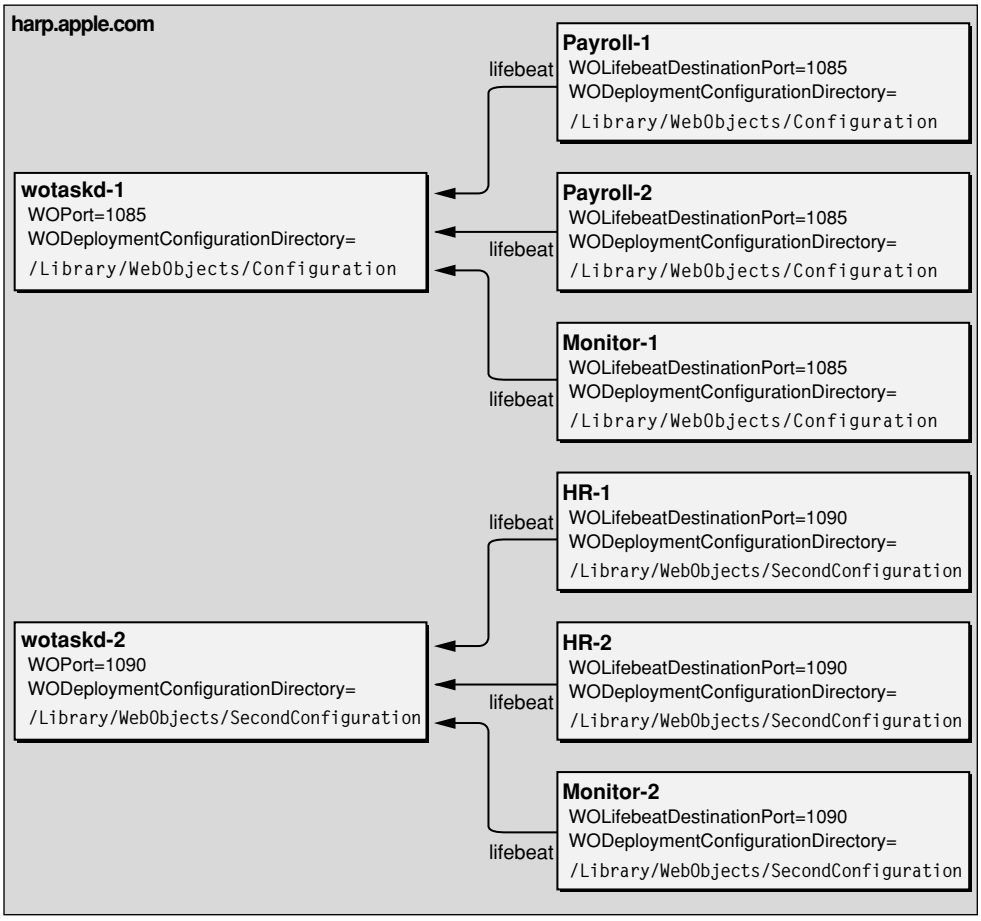
Deploying Multiple Sites

You can deploy and configure separate sites on a set of computers by running multiple web servers, each with its own HTTP adaptor. Such a deployment requires a separate group of wotaskd processes running on the same port. You also need a separate JavaMonitor process to configure each site.

The default installation of the WebObjects Deployment package provides you with one site—one wotaskd process per host, running on port `1085`. To create a second site, using the same hardware, you have to add an additional wotaskd process to each of the hosts you want to use.

What separates the environments from each other are the `WOPort` and `WOLifebeatDestinationPort` settings of each wotaskd process and the configuration directory used for each site. The application instances send their lifebeats to their `WOLifebeatDestinationPort`, while wotaskd processes listen for them in their `WOPort`. Figure 5-21 illustrates two sites on one host.

Figure 5-21 Multiple application environments on one computer



Because JavaMonitor is not started by a wotaskd process, its `WOLifebeatDestinationPort` argument needs to be set to match wotaskd's `WOPort` setting.

For details on how to set the command-line argument values required when starting wotaskd and JavaMonitor processes for separate application environments, see `WOPort`, `WOLifebeatDestinationPort`, and `WODeploymentConfigurationDirectory` in *WebObjects Application Properties Reference*.

Application Administration

After deploying applications, you should monitor their performance to find out, among other things, whether you need to add instances to an application to improve response times. "Monitoring Activity" (page 81) shows you the kind of performance data you can collect about an application.

There are several steps you can take to improve your site's performance; most of them have been explained earlier. The section "Improving Performance" (page 88) contains a list of recommended measures that help optimize your site's operation.

Monitoring Activity

There are several ways to obtain information about the applications deployed on your site. You can

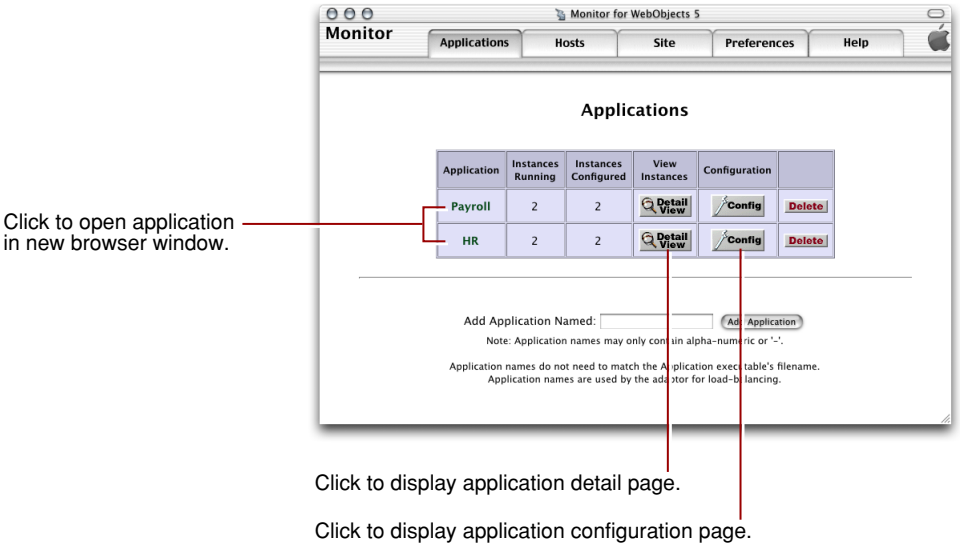
- use JavaMonitor
- analyze logs from application instances and HTTP adaptors
- view instance statistics webpages

Monitoring Application Performance

JavaMonitor's Applications page gives you an overall view of a site. Figure 6-1 illustrates the kind of information you can access through the Applications page:

- configured applications
- number of configured instances per application
- number of running instances per application

Figure 6-1 The Applications page



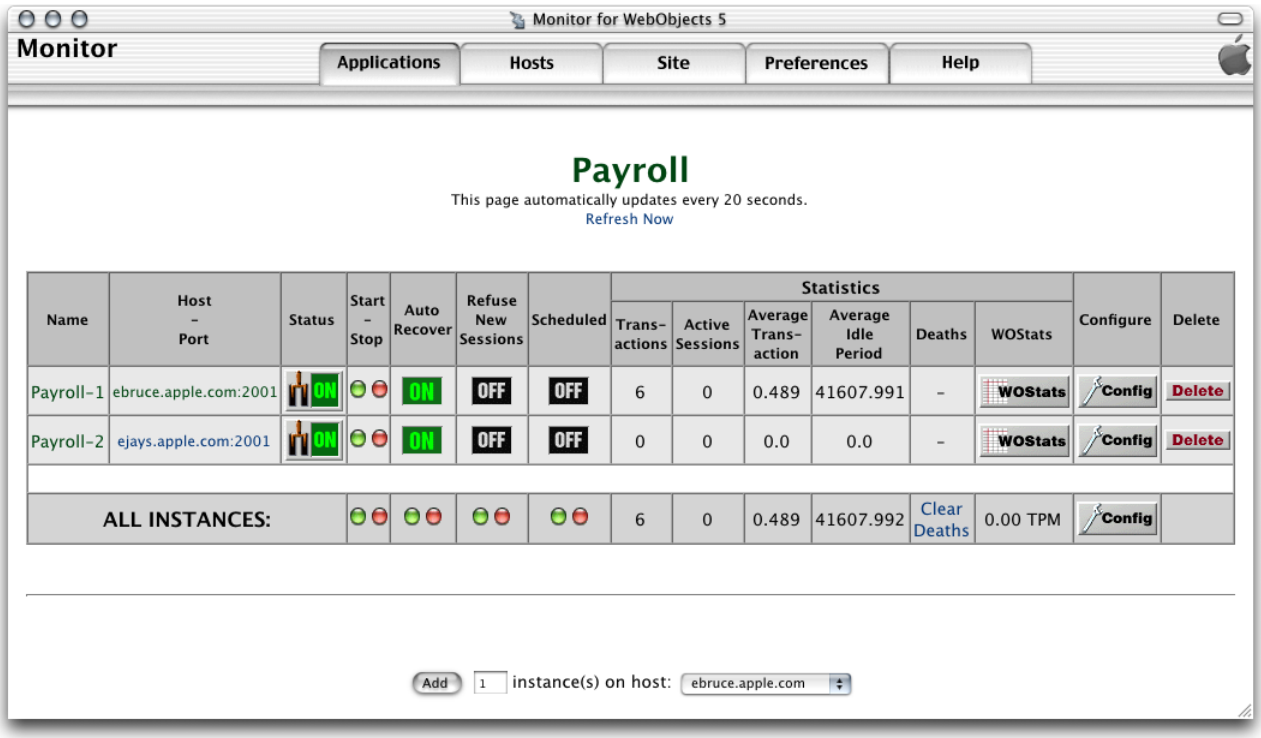
From this page, you can perform several tasks:

- **Connect to an instance of an application**. The entry page of the application is displayed in a new web browser window.
- **Display the application detail page of an application**. For details on the information provided, see "Adding an Application" (page 60).
- **Display the application configuration page**. See "Configuring an Application" (page 61) for more information.
- **Delete an application**. You confirm that you really want to delete the application (including all of its instances) in a confirmation page before the deletion takes place.

The Application Detail Page

Figure 6-2 depicts the application detail page.

Figure 6-2 The application detail page



The application's name is displayed centered and in bold letters. When there are active instances, it is a link to the application through the HTTP adaptor; the request is load balanced among the application hosts configured in the adaptor. (For this to work, the HTTP adaptor URL property must be set; see "Configuring Sites" (page 74) for details.) If no value has been entered for the property, the default URL (`http://localhost/cgi-bin/WebObjects/`) is used instead.) The URL used to connect to the application looks similar to the following:

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

The following list describes the performance-related information shown on the application detail page in the Statistics column group:

- **Transactions**: The number of requests the instance has received since it was started.
- **Active Sessions**: The number of active sessions (users) currently maintained by the instance.
- **Average Transaction**: Average time, in seconds, the instance has taken to process requests.
- **Average Idle Period**: The average time, in seconds, that the instance is idle (average time between requests).
- **Deaths**: The number of unexpected failures or deaths the instance has had since it was started. These exclude scheduled shutdowns or manual shutdowns through JavaMonitor.
- **WOStats**: Click the WOStats button to display the statistics page for the instance in a new web browser window. See "The Instance Statistics Page" (page 84) for details.

Before you can view the instance statistics page, you have to enter the password you set on the instance configuration page. See "Setting a Password for the Instance Statistics Page" (page 74) for details.

The row with the caption ALL INSTANCES contains application-wide performance information, including the average number of transactions processed per minute (TPM).

The Instance Statistics Page

You can access the instance statistics page of an instance through JavaMonitor or directly through a web browser.

To use JavaMonitor, go to the application detail page and click WOStats for the instance whose statistics you want to view.

To use your Web browser, access a URL similar to the following:

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

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

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

Figure 6-3 and Figure 6-4 show the information the instance statistics page provides.

Figure 6-3 The instance statistics page—part 1 of 2

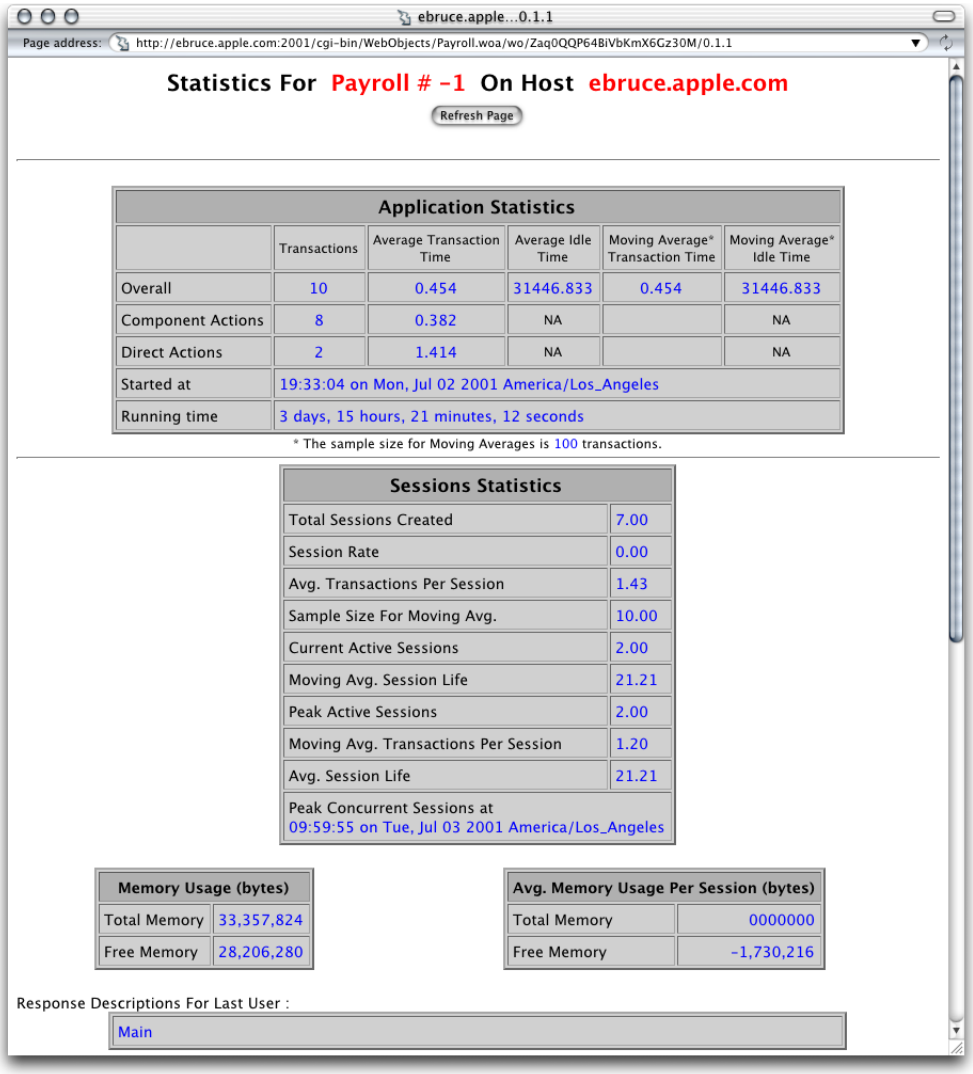
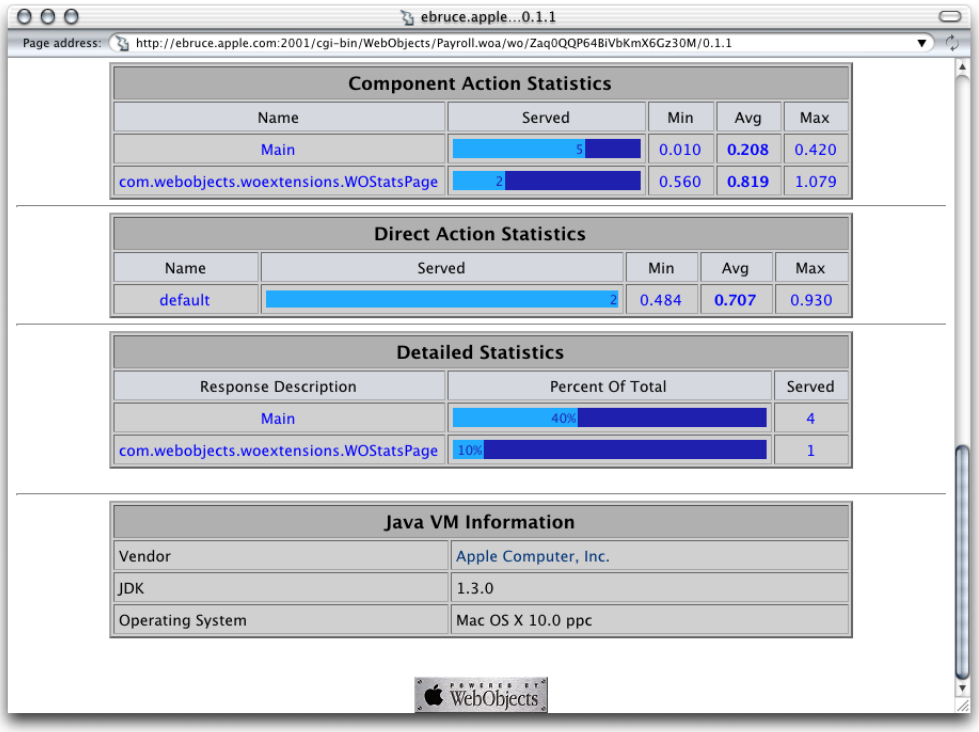


Figure 6-4 The instance statistics page—part 2 of 2



Logging and Analyzing Application Activity

WebObjects applications can log information that can be analyzed by a Common Log File Format (CLFF) standard analysis tool. Applications do not log their activities by default; logging must be enabled through an application's code. When enabled, the application records a list of components accessed during each session. By default, only component names are recorded, but you may add more information. An application's activity log is sent to the standard error stream.

Logging and Analyzing Adaptor Activity

To enable adaptor logging, you create a file called `logWebObjects` in the temporary directory of the computer where the web server runs. When logging is enabled, the adaptor logs its activity in a file called `WebObjects.log` in the temporary directory.

Note: Logging adaptor activity significantly decreases performance. Use this feature only as a troubleshooting aid; do not use it during regular deployment.

The location of the temporary directory depends on Mac OS X Server is `/tmp`.

Creating the Adaptor Log File

On UNIX-based platforms, do the following to create the `logWebObjects` file (you must have root privileges):

1. Start a command-shell window.
2. Set the working directory to the temporary directory.
3. Enter the following command:

```
touch logWebObjects
```

You can use the `tail` (UNIX) command to display the adaptor log file in your console.

On UNIX-based systems, use the following:

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

Analyzing the Adaptor Log File's Contents

You can analyze the information in the log to find out such things as which applications are being requested, which applications are being auto-started, and what the HTTP headers of the requests are. You can also use the log to verify that the HTTP adaptor is properly configured for load balancing.

The following excerpt includes an error message that indicates that an instance of Payroll wasn't running when a request for it came in:

```
Info: <WebObjects Apache Module> new request: /cgi-bin/WebObjects/Payroll
Debug: App Name: Payroll (7)
Info: Specific instance Payroll: not found. Reloading config.
```

After Payroll is started, the same request produces the following log entries:

```
Info: New response: HTTP/1.0 200 Apple WebObjects
Info: ac_newInstance(): added Payroll:2 (2001)
Info: V4 URL: /cgi-bin/WebObjects/Payroll
Info: loadaverage: selected instance at index 4
Info: Selected new app instance at index 4
Debug: Composed URL to '/cgi-bin/WebObjects/Payroll.woa/1'
Info: New request is GET /cgi-bin/WebObjects/Payroll.woa/1 HTTP/1.0

Info: Sending request to instance number 1, port 2001
Info: Trying to contact Payroll:1 on (2001)
Info: attempting to connect to ebruce.apple.com on port 2001
Info: Created new pooled connection [1] to ebruce.apple.com:2001
Info: Using pooled connection to ebruce.apple.com:2001
Info: Payroll:1 on (2001) connected [pooled: Yes]
Info: Request GET /cgi-bin/WebObjects/Payroll.woa/1 HTTP/1.0
 sent, awaiting response
Debug: ac_readConfiguration(): skipped reading config
Info: New response: HTTP/1.0 200 Apple WebObjects
Info: Payroll 1 load avg = 1
Info: received ->200 Apple
```

Improving Performance

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

- Configure your operating system so that it delivers the highest performance for your needs. Consult your operating system's documentation and your web server's documentation for performance-tuning information.
- When possible, use an API-based HTTP adaptor instead of a CGI adaptor.
- Make sure that the applications are written to perform optimally.

  The WebObjects developer documentation contains coding suggestions that help improve the performance of WebObjects applications.
- Enable component-definition caching for all applications.

  When applications are deployed, component-definition caching should be enabled so that each component's HTML and declarations files are parsed only once per session.
- Shut down and restart application instances periodically.

  Scheduling instances to shut down and restart periodically increases application performance and reliability by reducing the effects of memory leaks. For more on scheduling, see "Scheduling" (page 66).

  If your applications use custom scheduling algorithms to shut down, you should not use JavaMonitor's scheduling feature. Instead, use JavaMonitor's auto-recover feature. For more information, see information about Auto Recover in "New Instance Defaults" (page 62).
- Consider changing the physical configuration of your system.

  Determine the size of a single application instance (you can find this data on the application's instance statistics page) and multiply it by the number of instances you intend to run on a given computer. (For information on the instance statistics page, see "The Instance Statistics Page" (page 84).) The result is the amount of physical memory needed for that application. You have to add the memory required by the operating system, web server, and any other applications that run constantly on the computer. The result is the amount of physical memory that should be installed on the computer.
- Try to reduce the size of the application instance by limiting the amount of state that it stores. Set the session timeout value to ensure that sessions expire after a reasonable length of time. See `WOSessionTimeOut` in *WebObjects Application Properties Reference* for details on setting the session timeout interval for application instances.
- Make sure that the web server serves all static content, not your application.

If you use WebObjects Deployment mainly to deploy applications that access a data store, you achieve the best performance with a dedicated data-store server and a separate server for WebObjects applications.

Application URLs

An application URL with a specific format is used to make an initial request to a WebObjects application. The host identified in the URL receives the request and passes it to the WebObjects adaptor identified in the URL. The adaptor receives the request, repackages it into a standard format, and sends it to the appropriate WebObjects application instance. This section describes the format of this URL so that you can connect directly to an application instance from a browser. This URL is very useful for both development and deployment.

Types of WebObjects URLs

There are three ways to connect to an application instance:

- Connect directly to an application instance. The browser sends requests and receives responses directly from the application—the web server is not involved. The browser is connected directly to the port used by the application. Application resources—such as images—are served by the application. This approach does not require use of wotaskd or JavaMonitor.

  This is an example of a direct connect URL:

  ```
  http://localhost:2001/cgi-bin/WebObjects/MyApp.woa
  ```

- Connect through the web server to a deployed application. The browser communicates with the web server typically using port `80`. The web server passes requests to the WebObjects HTTP adaptor, which passes it to the application. The application sends responses to the adaptor, the adaptor sends them to the web server, and the web server sends them to the browser. Application resources are served by the web server. Therefore, this approach requires a split installation.

  This is an example of a deployed URL:

  ```
  http://host/cgi-bin/WebObjects/MyApp.woa
  ```

- Connect through the web server to a development application—an application that is running but not deployed. This approach is similar to accessing a deployed application except that load-balancing between instances does not occur. Use this approach to test an application through the web server without fully deploying it.

  This is an example of a development URL:

  ```
  http://host/cgi-bin/WebObjects/MyApp.woa/-2001
  ```

Format of WebObjects URLs

The format of a WebObjects application URL depends on the type of URL connection.

The URL format for an application in direct connect mode is:

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

The URL format for an application in deployment mode is:

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

The URL format for an application in development mode is:

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

Table 7-1 describes the variables in these URL formats.

Table 7-1 WebObjects application URL variables

Variable	Description
`host`	The host name of your computer or `localhost`.
`port`	The port number. If you are connecting directly to the application instance, include the port number.
`cgi-bin`	The `cgi-bin` directory of your server, usually `cgi-bin` or `Scripts`.
`WebObjects`	The name of the CGI adaptor, usually `WebObjects`.
`App`	The application name. The `WOApplicationBaseURL` property described in *WebObjects Application Properties Reference* specifies 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. See Table 7-2 for component actions and Table 7-3 for direct actions.

The two main request handlers are `WOComponentRequestHandler` and `WODirectActionRequestHandler` for component action URLs. The rest of the format for component action URLs is:

```
componentName / sessionID / elementID
```

Table 7-2 describes the variables in a component action URL.

Table 7-2 Component action URL variables

Variable	Description
`componentName`	Name of the WOComponent class.
`sessionID`	The session identifier.
`elementID`	The element identifier.

`WODirectActionRequestHandler` handles direct actions. The rest of the format for direct action URLs is:

```
[ actionClass | actionName | actionClass / actionName ][? key = value &amp;amp; key
 = value .....]
```

Table 7-3 describes the variables in a direct action URL.

Table 7-3 Direct action URL variables

Variable	Description
`actionClass`	The name of the WODirectAction class in which the action method is defined. If actionClass is not specified, class DirectAction is assumed.
`actionName`	The name of the action method. If `actionName` is not specified, the `defaultAction` method on `actionClass` is assumed.
`key`	The key portion of a key-value pair in the URL query string
`value`	The value portion of a key-value pair in the URL query string.

JMX Monitoring

Introduction

Important: The WOStatisticsStore MBean Interface is available in WebObjects 5.4 and later. JMX is available on computers with Sun's Java JDK 1.5 or later installed.

You can monitor WebObjects applications using Sun's Java JDK 1.5 Java Management Extensions (JMX) management and monitoring tools. Specifically, you use Sun's `jconsole` tool to collect statistics on running WebObjects applications. This appendix explains how to set the Java virtual machine (JVM) system properties for local and remote monitoring of a WebObjects application and how to view the WebObjects statistics with JMX.

This appendix provides only a brief introduction to JMX. They can be an alternative solution for monitoring your applications instead of using JavaMonitor and can be useful for tracing memory and thread usage during development. See Sun's Using the Platform MBean Server and Platform MBeans for more information on accessing Sun's MBeans.

Enabling JMX Monitoring

To enable the JMX agent and use `jconsole` to configure its operation, you must first set some system properties when you start the JVM. Note that properties specified on the command line override properties specified in a configuration file. You can either monitor a WebObjects application locally as described in "Local Monitoring" or remotely as described in "Remote Monitoring" (page 94). Local monitoring is useful in a development environment, but not for deployment, since running `jconsole` is very resource intensive.

Local Monitoring

To enable JMX local monitoring, add the property either via the command line or in the `build-properties.xml` file of the WebObjects application. For reference, see a sample `build-properties.xml` in the `/Developer/Examples/JavaWebObjects/HelloWorld` folder.

From the command line, set the `com.sun.management.jmxremote` property to `true` when starting the WebObjects application as follows:

```
cd MyApplication.woa

./MyApplication -Dcom.sun.management.jmxremote=true
```

To set this property in the `build-properties.xml` file located in your project directory, open the file using a text editor, and modify it as follows:

```
<jvmopts>-Dcom.sun.management.jmxremote=true</jvmopts>
```

To turn off JMX monitoring, make sure that you do not define the property anywhere in your application. For example, if the property is set in the `build-properties.xml` file, you should delete the definition of the property in that file.

Note: Local monitoring is useful for development or prototyping, but you should use remote monitoring during deployment, because `jconsole` consumes significant system resources. Remote monitoring isolates `jconsole` from the WebObjects application being monitored.

Remote Monitoring

To enable JMX remote monitoring, use either the command line or add the following JVM properties to the `build-properties.xml` file of the WebObjects application.

From the command line, set the port number for `com.sun.management.jmxremote.port` to some unused port number. Setting the port number automatically enables JMX monitoring. In a development environment, you can turn off authentication by setting the `com.sun.management.jmxremote.authenticate` and `com.sun.management.jmxremote.ssl` properties to `false`.

To set these properties in the `build-properties.xml` file for remote monitoring in a development environment, set the following JVM properties as follows:

```
<jvmopts>-Dcom.sun.management.jmxremote.port=9000</jvmopts>
<jvmopts>-Dcom.sun.management.jmxremote.authenticate=false</jvmopts>
<jvmopts>-Dcom.sun.management.jmxremote.ssl=false</jvmopts>
```

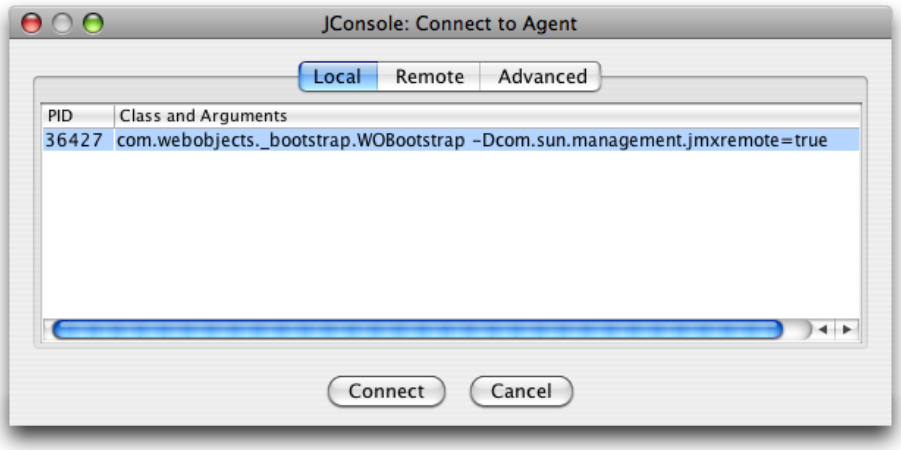
Warning: This sample configuration is insecure. Any remote user who knows (or guesses) your JMX port number and host name is able to monitor and control your WebObjects application and platform. While this configuration is OK for development, it is not recommended for deployment. See Sun's Monitoring and Management using JMX for how to set up SSL authentication for secure monitoring of your applications.

Using jconsole

After enabling JMX monitoring, you can view various measurements about the performance and resource consumption of your WebObjects application using Sun's `jconsole` tool.

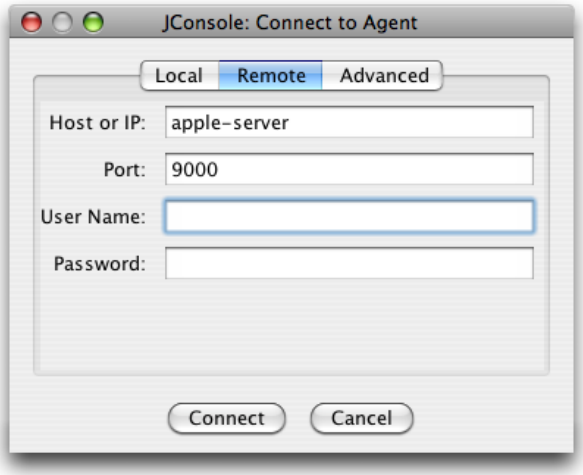
To monitor locally, start `jconsole` in a terminal shell and select the WebObjects application process ID (PID) as shown in Figure A-1.

Figure A-1 Local monitoring



To monitor remotely, start your WebObjects application on the remote machine with the proper JVM properties set as described in "Remote Monitoring" (page 94). Start `jconsole` in a terminal shell on a local machine and select the Remote tab as shown in Figure A-2 (page 95). Enter the host name and port number for the remote machine, and a user name and password if password authentication is enabled.

Figure A-2 Remote monitoring



Note: For more information on using `jconsole`, see Sun's Using JConsole.

Viewing WebObjects Statistics

You can define what to monitor by declaring Java standard MBean interfaces for your Java classes. Whenever you want to monitor your object, you register your object with the MBean server. Similarly, you can unregister your object with the MBean server whenever you are no longer interested in monitoring your object.

To accomplish these tasks, use the following WOApplication methods:

```
public MBeanServer getMBeanServer() throws IllegalAccessException
public String getJMXDomain()
public void registerMBean(Object aMBean, ObjectName aName)
public void registerMBean(Object aMBean, String aDomainName, String  aMBeanName)
 throws IllegalArgumentException
public void unregisterMBean(ObjectName aName)
```

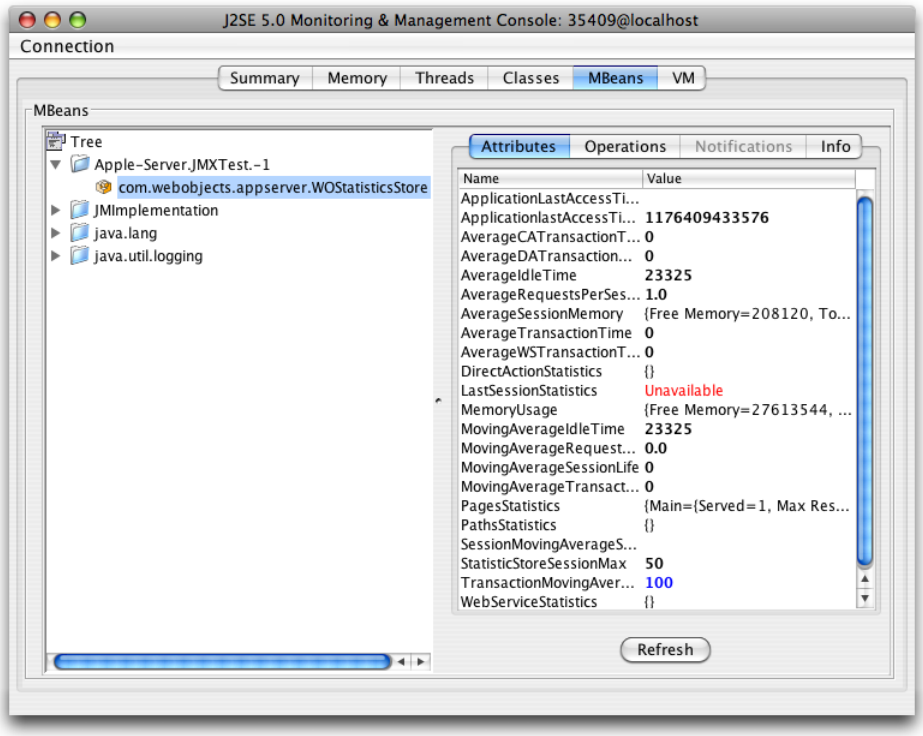
For example, registering your application with WOStatisticsStore MBean server gives you a view of various runtime statistics using `jconsole`. The WOStatisticsStore object records the bulk of its statistics at the end of each cycle of the request-response loop.

Listing A-1 shows how to register your WebObjects application with WOStatisticsStore MBean. To view the WebObjects statistics, start your application, launch `jconsole`, and select the MBeans tab to view various information as shown in Figure A-3 (page 97).

Listing A-1 Registering your application with the WOStatisticsStore MBean

```
// JMX API
import javax.management.MBeanServer;
public class Application extends WOApplication {
    public static void main(String argv[]) {
        WOApplication.main(argv, Application.class);
    }
    public Application() {
        super();
        System.out.println("Welcome to " + this.name() + "!");
        /* ** Put your application initialization code here ** */
        try {
            MBeanServer server = getMBeanServer();
          String mbeanClassName = "com.webobjects.appserver.WOStatisticsStore";
            String mbeanObjectNameStr = getJMXDomain() + ":type=" +
mbeanClassName;
            registerMBean((Object)statisticsStore(), getJMXDomain(),
mbeanClassName);
        } catch (Exception iae) {
            NSLog.debug.appendln("WARNING: couldn't initialize JMX monitoring
due to "+iae.getMessage()+"\n");
            iae.printStackTrace();
        }
    }
}
```

Figure A-3 Viewing WebObjects statistics



Deployment Issues with Java Client

Important: Java Client is deprecated and no longer supported in WebObjects 5.4 and later.

This chapter addresses special issues you need to address when deploying WebObjects applications using Java Client.

A WebObjects application developer can produce applications of two types:

- **Web applications** on which the user interface elements are produced using HTML code.
- **Java Client applications**, which use Sun's Swing technology to produce a user interface that is more appealing and more efficient than HTML-based interfaces. For more information on Java Client applications see the WebObjects Java Client documentation, available at http://www.apple.com/developer.

There are two main issues that you should keep in mind when you deploy and administer Java Client applications:

- **Session timeout**

  Java Client applications offer the user an interface that is very similar to the one offered by regular desktop applications. Therefore, users expect Java Client applications to behave in a way similar to their desktop applications.

  One of the main differences between a desktop application and a Java Client application is that Java Client applications open a connection to a server-side application. This connection expires after a certain period of inactivity. By default, the timeout period is 30 minutes. This may not be enough time for an application user that launches the application, goes to lunch, and returns to work 45 minutes later. When the user tries to use the application (which is still running on her computer), she will see a dialog that indicates that her session has timed out. In addition, any changes that were not saved are lost.

- **Traffic level**

  In Web applications, the packets sent between the web browser and the web server tend to be large. However, the size of the packets doesn't vary much (the server always sends the entire page to the browser). In Java Client applications, the packets sent by the server during application startup can be large (the entire application or part of it is downloaded); subsequent packets are relatively small (user-entered data and search results, for example).

For more information on WebObjects's Java Client technology, refer to *WebObjects Java Client Programming Guide*

Glossary

API-based adaptor HTTP adaptor based on a programming interface specific to a particular web server. It allows CGI-like tasks to run as part of the main server process, avoiding the creation and termination of a process for each request.

application host A computer capable of running application instances.

bundle In Mac OS X systems, a bundle is a directory in the file system that stores executable code and the software resources related to that code. The bundle directory, in essence, groups a set of resources in a discrete package.

CGI (Common Gateway Interface) A standard for communication between external applications and information servers, such as web servers.

CGI adaptor HTTP adaptor that uses the Common Gateway Interface (CGI) to translate requests from an web server into requests to an application instance, and responses from an application instance to responses to the web server. The web server creates a CGI process to handle each request.

data-source adaptor A mechanism that connects your application to a particular database server. For each type of server you use, you need a separate adaptor. WebObjects provides an adaptor for databases conforming to JDBC.

framework A type of bundle that packages a dynamic shared library with the resources that the library requires, including header files and reference documentation.

HTTP (Hypertext Transfer Protocol) The client-server TCP/IP protocol used on the web for the exchange of HTML documents.

HTTP adaptor A process (or a part of one) that connects WebObjects applications to an web server.

Java Client A WebObjects development approach that allows you to create graphical user interface applications that run on the user's computer and communicate with a WebObjects server application.

JDBC An interface between Java platforms and databases.

JDBC adaptor A datasource adaptor that allows WebObjects applications to connect to JDBC-compliant database management systems.

lifebeat Status message sent by WebObjects applications to wotaskd to report their activity. The four types of lifebeat messages are has started, is alive, will stop, and will crash.

load balancing Technique used to distribute user-load among the instances of an application. When multiple instances of an application are running and a new user accesses the application, the WebObjects adaptor uses one of several algorithms to determine which instance to forward the request to.

loopback Mechanism that allows you to open a connection to a computer that does not go over the network.

JavaMonitor A tool used to configure and maintain deployed WebObjects applications capable of handling multiple applications, application instances, and applications hosts at the same time.

session A period during which access to a WebObjects application and its resources is granted to a particular client (typically a browser). Also an object (of the WOSession class) representing a session.

SMTP (Simple Mail Transfer Protocol) A protocol used to transfer email between computers, usually over Ethernet.

socket Mechanism for creating a virtual connection between processes. It interfaces standard I/O with network communication facilities. A socket address consists of a port number and an IP address.

UDP (User Datagram Protocol) Lightweight and efficient connectionless datagram transport protocol. Used to send self-routing data throughout a network.

Web component An object (of the WOComponent class) that represents a webpage or a reusable portion of one.

Web server An application that serves webpages to web browsers using the HTTP protocol. In WebObjects, the web server lies between the browser and a WebObjects application. When the web server receives a request from a browser, it passes the request to the WebObjects adaptor, which generates a response and returns it to the web server. The web server then sends the response to the browser.

WebObjects Deployment Software package that allows you to deploy WebObjects applications on an intranet or the web. You need to install a WebObjects deployment license on computers on which you want to install this package.

WebObjects Development Software package that allows you to develop WebObjects applications. It includes tools to design applications using an object-oriented approach. You need to install a WebObjects development license on computers on which you want to develop applications.

WOServices WebObjects service that monitors wotaskd processes. Its main duty is to monitor wotaskd and restart it if it dies or when the host is restarted. The implementation of this service is platform-dependent.

wotaskd (WebObjects task daemon) WebObjects Deployment tool that manages the instances on an application host. It's used by Monitor to propagate site configuration changes throughout the site's application hosts.

Xcode A tool used to manage the development of a WebObjects application or framework.

Document Revision History

This table describes the changes to *WebObjects Deployment Guide Using JavaMonitor*.

Date	Notes
2007-10-31	Updated instructions for installing software when using Apache 2.
2007-07-24	Updated per WebObjects 5.4. Changed the title from "WebObjects Deployment Guide."
2005-07-07	Updated to reflect changes and new features in WebObjects 5.3. Changed the title from "Deploying Applications."
2005-06-04	Updated all references to Xcode.
2005-04-29	Updated wotaskd and Monitor information for WebObjects 5.2.4 and Mac OS X Server v.10.4.
2003-04-01	Added better explanation for split installations and multi-computer deployments to "Installing Applications" (page 58).
	Corrected misspelling of `logLevel` in "Setting WebObjects Options" (page 44).
2002-12-01	Made editorial changes.
2002-11-01	Revised for WebObjects 5.2.
2001-11-01	Changed conceptual images.
	Added information on `WODirectConnectEnabled` command-line argument.
	Changed references to `obj.conf` to `magnus.conf`.
	Added information on how and when the `SiteConfig.xml` file is written.
	Changed application settings screen shot to reflect the addition of the Project Search Path, Session Timeout, and Statistics Page Password text input fields.
	Added index.

Document Revision History