

# Project Wonder Deployment Guide

by Klaus Berkling



# INTRODUCTION

---

There are several ways to deploy your Project Wonder application.<sup>1</sup> It depends on the way your application was built, traffic and usage considerations. In the following section we will cover how to

- build your application for deployment,
- we'll look at deployment options based on usage,
- scaling,
- use tools to configure your site,
- deployment steps

Every deployment is different due to the application(s) usage patterns. The ideal deployment configuration will not be the one you first set up but it will evolve over time. This chapter may at times be vague - the best configuration is the one that meets your needs.

## OVERVIEW

---

### Build Considerations

There are two ways to build your application, embedding frameworks into your application or not.

Embedding your frameworks means all your frameworks are included in the application bundle. This includes third-party frameworks as well as Project Wonder, WebObjects and third party frameworks. 'Embedded' applications are usually much larger in size.

If you do not make embedded builds of your application all third-party frameworks, Project Wonder, and WebObjects frameworks need to be pre-installed on all deployment hosts.

Both have their benefits and drawbacks. Using 'embedded' applications allows for simpler host setups because it eliminates complicated class path configurations. Also, if you plan to run several different applications using embedded builds allows you to use different versions of your frameworks in those application. You avoid issues if the same framework is compatible with one application but not another.

If you plan to deploy a single application to a number of hosts, it might be more efficient to keep your application small and keep the static frameworks on each of your hosts.

Typical deployments are single host setups with several different applications with their frameworks embedded.

### Deployment Actors & Data Flow

There are several parts to a Project Wonder site that play a role in the data flow when a request comes in to your web application.

#### Deployment Actors

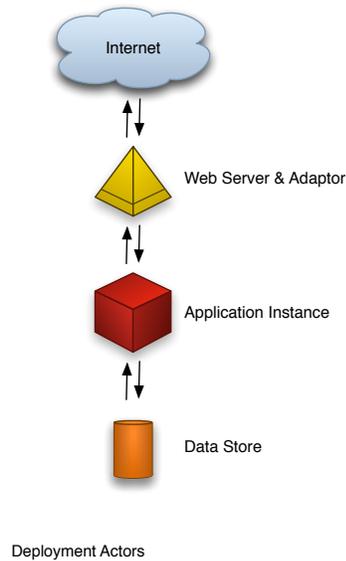
There are several parts which make up every site: the client, the web server, adaptor module or CGI, application instances, and the data store.

A *client* which can be a web browser or other application which consumes services provided by your web application.

A *web server*, usually Apache.

---

<sup>1</sup> The term application or web application in this context refers to a Wonder application.



web server and application instance host(s). Figure 2 shows a multi-host configuration.

## The Request/Response Loop

The request/response loop is the path a request makes starting as incoming URL to the web server forwarded on to the application instance. The response is constructed in the application and sent back in the form of HTML, JSON, XML, or other formatted data. If the application instance does not have the requested data in its cache it will retrieve it from the data store.

The HTTP *adaptor* usually loaded by Apache as a module or run as a CGI. The adaptor communicates with application instances.

An *applications instance*. Each running copy of an application is called an application instance.

A *database store* such as PostgreSQL, MySQL, or Oracle. Project Wonder supports several database servers. A third party JDBC connection is required which can be embedded in your application or installed on your hosts.

The web server, application instances, and data store can be deployed independently from each other. They can be deployed on one host or separately on multiple hosts. When to use multiple hosts depends on traffic load and redundancy requirements. Figure 1 is a single host configuration. Here all actors are on one host. This is a good starting point and make excellent test platforms. It may be beneficial for data-heavy sites to separate the database server from the

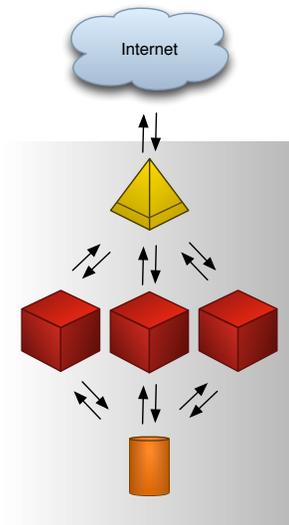


Figure 1

## Scaling

### Connection Threads & Instances

Scaling a site happens in two places, threading in the application and running several instances of the application on the deployment host(s).

Having many threads is useful if your application makes frequent use of cached data. All threads make use of the same cache. Additional application instance will have their own cache. More threads means fewer instances. An application instance usually has only one database connection, any request requiring data access is queued with other database requests.

Running multiple application instances is useful when serving unique content for individual users. Each application instance gives you an additional concurrent connection to your database<sup>2</sup>. Running multiple instances of an application is more of a scaling option.

When a site runs multiple instances an issue with 'stale' data might become apparent. Updated data is available to all sessions of an individual instance. Other application instances will only get the updated data once the cached enterprise objects expire and are refetched from the database.

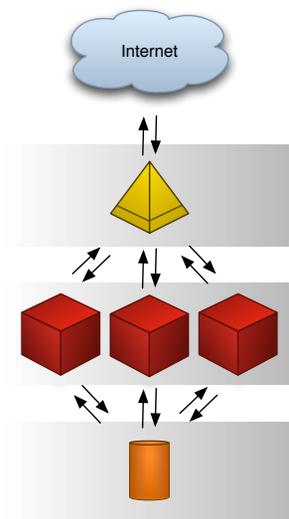


Figure 2

<sup>2</sup> This is not always true. There are several smart ways using the editing context to open multiple database connection. Discussing editing contexts is beyond the scope of this document.

An application which shows the same information to a large number of users can take advantage of the instance cache. An application that mostly shows users' individual data may require more instances because the data must be retrieved from the database store.

## Instance Hosts

The number of instances largely depends on RAM usage and availability. Each instance runs its own Java VM. The number of instances and Java VM RAM allocation can be adjusted. Eventually a host is running all the instances it can. Once this point is reached additional instance hosts can be added. See figure 3, an arbitrary number of instance hosts can be added to the setup.

Each actor in the site can be separately scaled depending on its load capacity. The only way to properly scale the actors is to monitor each actor and address each separately. There is a typical path to scaling though:

- a. Adding instances. The number of concurrent connections in a web server can easily exceed the number of connection threads in an application instance.
- b. Adding instance hosts. This can potentially double the load capacity. At this point you can consider adding another web server so that each instance host has its own web server. This is a smart way to achieve redundancy, however a proper load-balancing solution needs to be utilized (round-robin DNS may suffice, it's inexpensive but lacks fail-over).
- c. Add another web server. Eventually the whole site needs to be split into two clusters<sup>3</sup>. It is not possible that two web server hosts send their requests to the same set of instance hosts. An adequate load-balancing system needs to be put in front of the web servers.
- d. There is a direct link between the number of instances and database connections. Usually two connections are required per application instance, although only one is used. You need to be able to scale up your database along with increased application instances.

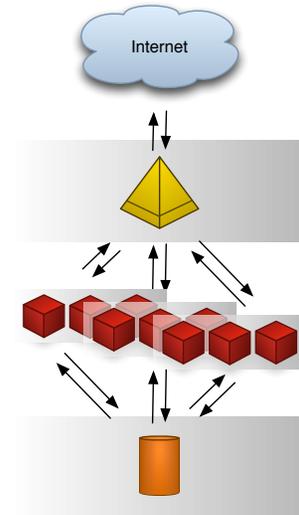


Figure 3

In most situations, multiple application instances will meet your needs. Requirement for additional instance hosts, web servers and database capacity, in this order, happens much later.

# SETUP DETAILS

## Site Configuration

### Deployment and management tools

The deployment and management tools consist of JavaMonitor and wotaskd or WO Task daemon.

**JavaMonitor** is a Project Wonder application which has its roots in WebObjects JavaMonitor. There have been bug fixes and many new features have been added. JavaMonitor is a Project Wonder application that allows you to add your application to your site and configure its parameters. JavaMonitor also allows you to add hosts that run application instances and to designate which of those run your application instance(s).

JavaMonitor should only be running on one host in your cluster, typically your web server host. JavaMonitor reads and writes the site's config file which it synchronizes with all instance hosts in your cluster<sup>4</sup>.

<sup>3</sup> In this context a cluster is a collection of interconnected servers that work together in some fashion.

<sup>4</sup> A site may consist of one or more clusters.

JavaMonitor is capable of managing Mac OS X, Windows, and UNIX-type instance hosts concurrently. You can run a mis-match of operating systems in your site.

**wotaskd** is another Project Wonder application which also began as a WebObjects application. It behaves as a daemon and must be running on all hosts making up your site. A wotaskd daemon manages the instances running on its host. It executes schedules and makes sure instances are running should they stop unexpectedly<sup>5</sup>.

Both JavaMonitor and wotaskd have REST routes to automate deployment tasks and for monitoring a site.

## Configuring A New Application

Adding a new application to a site is relatively simple. JavaMonitor has the facility to add applications and deployment parameters such as:

- New Instance Defaults
- Application Settings
- Scheduling
- Email Notification
- Instance Load Balancing and Adaptor Settings.

### New Instance Defaults

**Path:** This is the path to the startup script of your web application. Your application bundle will contain two scripts to launch your application. One UNIX and Mac OS X shell script and one Windows .cmd script. Click the Pick button to select the right script for your platform.

Usually applications are installed in /Library/WebObjects/Applications. On some UNIX servers applications might be installed in /opt/Local/Library/WebObjects/Applications. If you have different paths for your application because you run different operating systems, you can select separate paths for each OS. Note: the paths on each OS-equivalent host must be the same.

**Auto Recover:** This is an option that causes newly added application instances to start up automatically just after they have been added. This may not be desired in situations when instances are added by other tools. Crashed instances are also automatically restarted if this option is enabled.

**Minimum Active Sessions:** There are situations where an instance will terminate itself when this minimum number of instances is reached (usually when scheduled). Usually this value is zero.

**Cache enabled:** Uncertain, usually enabled.

**Debugging enabled:** Uncertain, usually disabled.

**Output Path:** Path to the log file. Select a directory which should contain the logs, terminate it with a directory separator ( / for UNIX, \ for Windows). The user under which the application instances is running should have permissions to write to this directory. Usually the user is appserver and the group is appserveradm.

**Auto Open in Browser:** When enabled, your default web browser will open to the application instance. This is usually disabled.

**Additional Argument:** These are any command line arguments appended to the startup command script. Several site specific variables can be set here. See your applications Properties file for available options. Command line argument override those in your application's Property file and all other Property files.

**Lifebeat Interval:** This is used by wotaskd. The wotaskd daemon will check the life-beat thread in the application instance in this interval. If it fails it will assume the instance crashed and will attempt to restart it<sup>6</sup>.

---

<sup>5</sup> Instances are restarted if the auto-recover option is enabled.

<sup>6</sup> This feature will not help if your application instance has a hung thread, an immortal session or is dead-locked. The lifebeat thread is very stable by itself.

Each of the options above can be picked up by new instances or can separately be pushed to existing instances<sup>7</sup>. Usually these options are set before any instances are added and later pushed to all instances together when changed. Later we'll see how instances can be configured individually for debugging purposes.

## Application Settings

**Name:** Allows the application name to be changed. It does not have to be the project name and does not have to end with .woa.

**Starting Port:** All instances are like mini web servers, they all require a unique port to accept request. Usually ports are assigned sequentially. This field allows you to set the starting port number.

**Time Allowed For Startup:** If you instance does not start up in the allotted time, it is considered dead and will get restarted.

**Phased Startup:** This option is only useful if you run very many instances of your application. It's keeps the load on your hosts to a minimum because each instance starts up individually. However, if you have many instances bringing your site up can take a long time. When your site receives traffic as instances are coming up you will see uneven traffic across your instances.

**Adaptor:** There is only one: WODefaultAdaptor.

**Minimum Adaptor threads:** The minimum number of threads your instance will run. This number should be relatively low, about 10. This number should be based on tests you run against an individual instance.

**Maximum Adaptor threads:** The maximum number of threads your instance will run. This number should be relatively low, about 20. This number should be based on tests you run against an individual instance. These numbers relate to connections your instance can handle. It also relates to the number of connections which could wait when accessing your database. Too many threads and too many connection could queue up to use the database.

**Adaptor threads:** Applies only to WebObjects 4.5.x users.

**Listen Queue Size:** The size of the connection queue before they get a thread in the instance. This number should be based on tests you run against an individual instance. This number in combination with max. adaptor threads is the number of connections your instance will accept. It's important that your application can handle these connections in timely fashion.

**Project Search Path:** Unknown.

**Session TimeOut (sec):** The time in seconds your instance sessions will time out. If your application is state-full, sessions will get terminated after this time.

**Statistics Page Password:** Each instance keeps track of certain statistics. To access this page set the password [here](#).

## Scheduling

**ID:** Every instance has a number, which also shows up in the log names for each instance.

**Is Scheduled:** If enabled the schedule is applied to this instance. This also enabled auto recover.

**Graceful Scheduling:** If enabled wotaskd tells the instance to to refuse new sessions. Once the min. active sessions value is reached the instance terminates itself. If this option is disabled, the instance is terminated regardless of active sessions.

---

<sup>7</sup> Existing instance will pick up the new command line arguments once they restart.

**Schedule Type:** The types of schedules are every given numbers of hours, once a day, or every week. Each type can be set to a certain time to take effect.

## Email Notification

This setting is only available when the site has been configured for a mail server and return email address.

**Email Notifications are Enabled:** If enabled, an email will be sent to the given recipient in case an instance dies.

## Load Balancing and Adaptor Settings.

These are parameters the Apache adaptor uses when communicating with the application instances:

**Load balancing scheme:** This is the method used by the Apache adaptor to distribute requests to application instances.

**Retries:** The number of attempts the adaptor tries to find a working application instance

**Redirection URL:** The URL the client is redirected to when no instances are available.

**Dormant:** The number of times an application instance is skipped over before trying it again.

**Send timeout:** Time in seconds the adaptor waits trying to send data to the application instance.

**Receive timeout:** Time in seconds the adaptor waits receiving data from the application instance.

**Connect timeout:** Time in seconds the adaptor waits establishing a connection to an application instance.

**Send Buffer Size:** Size of the TCP/IP socket send buffer.

**Receive Buffer Size:** Size of the TCP/IP socket receive buffer

**Connection pool size:** Number of connection kept open for each application instance.

**URL Version:** WebObjects version used for URL parsing. Should always be 4.

## Site Setup

A Project Wonder site consists of three major rolls: the web server, instances, and the data store. Setting up the data store server such as an Oracle or MySQL server is beyond the scope of this document. Installing Apache is also beyond the scope of this document. Other sources describe the Apache setup process in much better detail then can be done here.

### Setting Up WebObjects Hosts

The installation of a head node or instance node should be very similar or even identical. The difference is that an instance node does not run the web server. We'll use some commands from a set-up script used on Ubuntu (see the Files section for a full version of the script). You'll need root permissions to executes these commands.

Add the standard WebObjects group:

```
addgroup appserveradm
```

Add the standard WebObjects user with disabled login option:

```
adduser --disabled-login appserver
```

Create the WebObjects directories. This directory is going to contain Wonder applications, tools and some configuration files:

```
mkdir -p "/opt/Local/Library/WebObjects/Applications"
mkdir -p "/opt/Local/Library/WebObjects/JavaApplications"
mkdir -p "/opt/Local/Library/WebObjects/Adaptors/Apache2.2"
mkdir -p "/opt/Local/Library/WebObjects/Configuration"
mkdir -p "/opt/Local/Library/WebObjects/Logs"
```

Change permissions on the WebObjects directories:

```
chmod -R 755 "/opt/Local/Library/WebObjects"
```

Download Project Wonder's wotaskd and JavaMonitor to /opt/Local/Library/WebObjects/JavaApplications:

```
cd /opt/Local/Library/WebObjects/JavaApplications
curl -O "http://jenkins.wocommunity.org/job/Wonder/lastSuccessfulBuild/artifact/Root/\
Roots/wotaskd.tar.gz"
curl -O "http://jenkins.wocommunity.org/job/Wonder/lastSuccessfulBuild/artifact/Root/\
Roots/JavaMonitor.tar.gz"
```

Uncompress the downloaded files:

```
tar -zxf wotaskd.tar.gz
tar -zxf JavaMonitor.tar.gz
```

Download Project Wonder's Apache module to /opt/Local/Library/WebObjects/Adaptors/Apache2.2:

```
cd /opt/Local/Library/WebObjects/Adaptors/Apache2.2
curl -O "http://wocommunity.org/documents/tools/mod_WebObjects/Apache2.2/centos/5.5/\
x86_64/mod_WebObjects.so"
```

Set the owner and group on the the WebObjects files and directories:

```
chown -R appserver:appserveradm "/opt/Local/Library/WebObjects"
```

Add this to /etc/profile and /etc/environment. Some scripts make references to \$NEXT\_ROOT and thus needs to be defined.

```
NEXT_ROOT=/opt; export NEXT_ROOT
```

Add this to /etc/profile as well. This allows you to run JavaMonitor on port 8080 using the alias 'javamonitor':

```
alias javamonitor="sudo -u appserver '/opt/Local/Library/WebObjects/\
JavaApplications/JavaMonitor.woa/JavaMonitor -WOPort 8080'"
```

Add startup scripts for wotaskd. This launches wotaskd when the server starts up (see the Files section for a full version of the script):

```
cd /etc/init.d/
curl -O "http://www.berkling.us/wobootstrap/wotaskd"
update-rc.d wotaskd defaults
```

Optionally download a startup script for JavaMonitor (see the Files section for a full version of the script):

```
cd /etc/init.d/
curl -O "http://www.berkling.us/wobootstrap/womonitor"
update-rc.d womonitor defaults
```

Since we've made changes in the /etc/profile and /etc/environment files and also added new scripts in /etc/init.d we should restart the server.

## Configuring Apache

Apache, for the head node takes additional steps to configure.

Download the WebObjects Apache config file (see the Files section for a full version of the script):

```
cd /opt/Local/Library/WebObjects/Adaptors/Apache2.2
curl -O "http://www.berkling.us/wobootstrap/apache.conf"
```

Add this to the bottom of the Apache configuration file: /etc/apache2/httpd.conf:

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

Add this to Apache's main config file in the alias\_module section (This resolves issues with existing CGIs):

```
ScriptAliasMatch ^/cgi-bin/((?!(?i:webobjects)).*$) "/usr/lib/cgi-bin/$1"
```

Restart the web server for the changes to take affect.

## Application Deployment Steps

This section covers the deployment of a new Wonder application. We will assume that the server is already part of a working Wonder site.

A typical Wonder application has two parts: The application and web resources. If your application is called CustomerContacts and you use Eclipse IDE with WOLips, or Jenkins to build your application you will have two files:

- CustomerContacts-Application.woa.tgz, and
- CustomerContacts-WebResources.woa.tgz

## Stop Running Instances

This step is only necessary if you already have running instances.

1. **Startup JavaMonitor.** On UNIX OS's use the following command to run JavaMonitor:<sup>8</sup>

```
sudo -u appserver /opt/Local/Library/WebObjects/JavaApplications/JavaMonitor.woa/\
JavaMonitor -WOPort 8080
```

If JavaMonitor does not launch check the screen output for errors.

2. **Open JavaMonitor.**  
On a Mac OS X server JavaMonitor may automatically open your web browser

```
http://<your-hostname>:8080/cgi-bin/WebObjects/JavaMonitor.woa
```

3. **Stop running instances.**  
Navigate to your application and click View Details. On the new page of instances click the red bubble to stop the instance. If you have more than one instance you can use the master button at the bottom of the list.

At this point your Wonder application should be down.

## Installing Application Files

1. **Expand the ...-Application.woa.tgz archive to your WebObjects application folder.**  
The archive should expand to a .woa directory. You will need to designate a directory where all your Wonder applications are installed. For Mac OS X servers this is usually in "/Library/WebObjects/Applications". On Unix systems might have "/opt/Local/Library/WebObjects/Applications".
2. **Verify and change, if necessary, the owner and group of the .woa directory and all containing files and sub-directories.**  
The user and group usually used is appserver and appserveradm respectively  
(`chown -R appserver:appserveradm <path-to-application-directory>`)
3. **Verify the file permissions.**  
If your build process does not follow standard practices the file permissions may not be correct. All files and sub-directories should at least be readable for owner and group. All directories need their execution bit set for owner and group as well. Immediately in the .woa directory is a UNIX script that should have its execution bit set for owner and group.

Repeat this process for each host in your cluster.

Directory listing of a typical Wonder application where the application is named CustomerContacts:

```
drwxr-xr-x  9 _appserver  _appserveradm   306 Mar  4  2012 Contents
-rwxr-x---  1 _appserver  _appserveradm  11229 Mar  4  2012 CustomerContacts
-rw-r--r--  1 _appserver  _appserveradm  16374 Mar  4  2012 CustomerContacts.cmd
-rw-r--r--  1 _appserver  _appserveradm   6305 Mar  4  2012 WBootstrap.jar
```

## Installing Web Resources Files

Web resources are only installed on the web server of the WebObjects site. You can skip this step if you are updating an exiting application and you know there are no changes to web resources.

1. **Expand the ...-WebResources.woa.tgz archive into WebObjects folder in your web server documents root directory.**  
The archive should expand into a .woa directory. For Mac OS X servers this is usually on "/Library/WebServer/Documents/WebObjects". On other UNIX systems this is sometimes "/var/www".
2. **Verify and change, if necessary, the owner and group of the web resources directory.**  
The web resources files can be treated like any other web server files with the same owner, groups and permission as all other documents in the web server documents directory.

## Start Instances

<sup>8</sup> Mac OS X 10.5 was the last Mac OS version that supported WebObjects as part of the Server Admin tools. Always use Server Admin, if WebObjects is available.

If you previously stopped your instances you now need to start them. If you left them running you'll need to restart them.

1. **Startup JavaMonitor.** On UNIX OS's use the following command to run JavaMonitor:<sup>9</sup>

```
sudo -u appserver /opt/Local/Library/WebObjects/JavaApplications/JavaMonitor.woa/\
JavaMonitor -WOPort 8080
```

If JavaMonitor does not launch check the screen output for errors.

2. **Open JavaMonitor.**  
On a Mac OS X server JavaMonitor may automatically open your web browser

```
http://<your-hostname>:8080/cgi-bin/WebObjects/JavaMonitor.woa
```

3. **Start instances.**  
Navigate to your application and click View Details. On the new page of instances click the green bubble to start the instance. If you have more than one instance you can use the master button at the bottom of the list. You should see visual indicators that the instance is starting and later that it is on.

At this point your Wonder application should be running.

## Other Configuration Files

### Adaptor Configuration

The adaptor is an Apache module so it is configured through Apache's config files. Apache's config file at `/etc/apache2/httpd.conf` usually contains an include directive that reads in WebObjects additions to the Apache configuration. Usually this file is located at `/opt/Local/Library/WebObjects/Adaptors/Apache2.2/apache.conf` (omit `/opt/Local` for Mac OS X systems). A sample config file is available in the Files section of this document).

Depending on your deployment platform, Apache configurations can be located in different locations. Apache's 2.2 standard configurations is sometimes in `/etc/apache/apache.conf` and user additions are in `/etc/apache/httpd.conf`. Site specific configurations are sometimes located in `/etc/apache/sites-available` which have symbolic links in `/etc/apache/sites-enabled`. Only files in sites-enabled are ready in by Apache. Please review the Apache documentation to learn about how Apache is configured.

### Site Configuration

The site configuration file `SiteConfig.xml` in `/opt/Local/Library/WebObjects/Configuration` is an XML file that contains all the necessary information to run instances. It is read and synchronized by `wotaskd` between all hosts in the site. It should not be edited by hand.

## Files

The files and scripts reproduced in this section are for reference. They are taken from actual production sites but may not perform without errors on your particular host(s). Proper testing is strongly advised.

### Sample Apache configuration addition

`http://www.berkling.us/wobootstrap/apache.conf:`

```
# WebObjects 5.4: Enable the WebObjects module.
LoadModule WebObjects_module /opt/Local/Library/WebObjects/Adaptors/Apache2.2/mod_WebObjects.so

# Alternatively, You can load the WebObjects Module manually by typing and commenting out
# the previous LoadModule directive.
# /usr/bin/apxs -i -a -n WebObjects mod_WebObjects.la
#LoadModule WebObjects_module modules/mod_WebObjects.so

# Path to the Document Root of your Webserver,
# it should contain a directory named WebObjects
WebObjectsDocumentRoot /opt/Library/WebServer/Documents
```

<sup>9</sup> Mac OS X 10.5 was the last Mac OS version that supported WebObjects as part of the Server Admin tools.

```

# You can change the 'cgi-bin' part of WebObjectsAlias to whatever you
# prefer (such as Apps), but the 'WebObjects' part is required.
WebObjectsAlias /cgi-bin/WebObjects

# Here are the 3 possible configuration modes.
# The apache module uses one of them to get information
# about your deployed applications.
# 1085 is the reserved port on which wotaskd processes listen to by default.

# Host List Configuration
# wotaskd is started automatically on supported platforms,
# so this is the default mode.
# The apache module gets its configuration from the wotaskds
# listed on the configuration line
# For multiple hosts:
# WebObjectsConfig http://<a-host>:<port>,http://<another-host>:<port> <interval>
# For localhost:
WebObjectsConfig http://localhost:1085 10

# Multicast Configuration
# The apache module gets its configuration from all wotaskds
# that respond to the multicast call on the subnet
# WebObjectsConfig webobjects://239.128.14.2:1085 10

# File Configuration
# The apache module gets its configuration from one file
# WebObjectsConfig file://<path-to-a-xml-config-file> 10

# To enable public access to the WOAdaptorInfo page, uncomment the following line
#WebObjectsAdminUsername public

# To enable the WOAdaptorInfo page with restricted access,
# uncomment the next two lines and set the user and password
# To access the WOAdaptorInfo page with restricted access,
# use a URL like: http://webserver/cgi-bin/WebObjects/WOAdaptorInfo?user+password.
#WebObjectsAdminUsername monitor
#WebObjectsAdminPassword passwd

# To change the logging options, read the following comments:
# The option name is "WebObjectsLog" and the first value indicates the path of the log file.
# The second value indicates the log level. There are five, in decreasing informational order:
#   "Debug",   "Info",   "Warn",   "Error",   "User"
#
# Note: To enable logging, touch '/tmp/logWebObjects' as the administrator user (usually root).
#
# The following line is the default:
#WebObjectsLog /opt/Local/Library/WebObjects/Logs/adaptor.log Debug

```

## Sample Ubuntu boot-strap script

<http://www.berkling.us/wobootstrap/wo-setup4Ubuntu>

```

#!/bin/bash

# Wonder/WebObjects Deployment tools install script, from nothing

WD=`pwd`

USER="appserver"
GROUP="appserveradm"

NEXT_ROOT="/opt"
WEBOBJECTS="$NEXT_ROOT/Local/Library/WebObjects"

WOWEBROOT="/var/www/WebObjects"

echo "This script does not install Java, Apache, or your database engine."
echo "You are `whoami`. You need to be root to run this."

addgroup $GROUP

adduser --disabled-login $USER
adduser $USER $GROUP

echo "Creating WebObjects directory in $WOWEBROOT:"
mkdir "$WOWEBROOT"
chown -R $USER:$GROUP "$WOWEBROOT"

```

```
chmod -R 755 "$WOWEBROOT"

echo "Creating directories in $WEBOBJECTS"
mkdir -p "$WEBOBJECTS/Applications"
mkdir -p "$WEBOBJECTS/JavaApplications"
mkdir -p "$WEBOBJECTS/Adaptors/Apache2.2"
mkdir -p "$WEBOBJECTS/Configuration"
mkdir -p "$WEBOBJECTS/Logs"

chmod -R 755 "$NEXT_ROOT"

echo "Adding 'NEXT_ROOT=$NEXT_ROOT; export NEXT_ROOT' to /etc/environment"
echo 'export NEXT_ROOT=$NEXT_ROOT' >> /etc/environment
echo 'export NEXT_ROOT=$NEXT_ROOT' >> /etc/profile
echo 'alias javamonitor="sudo -u '$USER' '$WEBOBJECTS'/JavaApplications/JavaMonitor.woa/JavaMonitor -WOPort 8080"' >> /etc/profile

echo "Downloading Wonder wotaskd and JavaMonitor"
cd "$WEBOBJECTS/JavaApplications"
curl -O "http://jenkins.wocommunity.org/job/Wonder/lastSuccessfulBuild/artifact/Root/Roots/wotaskd.tar.gz"
curl -O "http://jenkins.wocommunity.org/job/Wonder/lastSuccessfulBuild/artifact/Root/Roots/JavaMonitor.tar.gz"
tar -zxf wotaskd.tar.gz
tar -zxf JavaMonitor.tar.gz

echo "Downloading Apache module"
cd "$WEBOBJECTS/Adaptors/Apache2.2"
curl -O "http://wocommunity.org/documents/tools/mod_WebObjects/Apache2.2/centos/5.5/x86_64/mod_WebObjects.so"
curl -O "http://www.berkling.us/wobootstrap/apache.conf"

echo "Changing ownerships on $WEBOBJECTS to $USER:$GROUP"
chown -R $USER:$GROUP "$WEBOBJECTS"

echo "Downloading launch scripts."
cd /etc/init.d
curl -O "http://www.berkling.us/wobootstrap/womonitor"
curl -O "http://www.berkling.us/wobootstrap/wotaskd"

chmod 755 womonitor
chmod 755 wotaskd

update-rc.d tomcat6 disable
update-rc.d wotaskd defaults

cd "$WD"

echo "Done. You may reboot your system"
```

## Sample Additions to Apache configuration file

<http://www.berkling.us/wobootstrap/httpd.conf-add>

```
# Replaces ScriptAlias to /cgi-bin in the alias_module section:
ScriptAliasMatch ^/cgi-bin/((?!(?i:webobjects)).*$) "/usr/lib/cgi-bin/$1"

...

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

## Sample Startup script for JavaMonitor on Ubuntu

<http://www.berkling.us/wobootstrap/womonitor>

```
#!/bin/bash

# chkconfig: - 90 20
# description: Provides WebObjects services

USER=appserver

# See how we were called.
case "$1" in
  start)
    echo -n "Starting Monitor: "
    export NEXT_ROOT=/opt
```

```

    su $USER -c "${NEXT_ROOT}/Local/Library/WebObjects/JavaApplications/JavaMonitor.woa/JavaMonitor -WOPort
8080 &"
    echo
    ;;
stop)
    echo -n "Shutting down Monitor: "
    MONITOR_PID=`ps aux | awk '/WOPort 8080/ && !/awk/ {print $2}`
    kill $MONITOR_PID
    echo
    ;;
restart)
    $0 stop
    $0 start
    ;;
*)
    echo -n "Usage: $0 {start|stop|restart}"
    exit 1
esac

if [ $# -gt 1 ]; then
    shift
    $0 $*
fi

exit 0

```

## Sample Startup script for wotaskd on Ubuntu

<http://www.berkling.us/wobootstrap/wotaskd>

```

#!/bin/bash

# chkconfig: - 90 20
# description: Provides WebObjects services

USER=appserver

# See how we were called.
case "$1" in
    start)
        echo -n "Starting wotaskd: "
        export NEXT_ROOT=/opt
        su $USER -c "${NEXT_ROOT}/Local/Library/WebObjects/JavaApplications/wotaskd.woa/wotaskd -WOPort 1085 \
-WOOutputPath $NEXT_ROOT/Library/WebObjects/Logs/WOTaskd.log -DWOTaskd.forceQuitTaskEnabled=true &"
        echo
        ;;
    stop)
        echo -n "Shutting down wotaskd: "
        WOTASKD_PID=`ps aux | awk '/WOPort 1085/ && !/awk/ {print $2}`
        kill $WOTASKD_PID
        echo
        ;;
    restart)
        $0 stop
        $0 start
        ;;
    *)
        echo -n "Usage: $0 {start|stop|restart}"
        exit 1
esac

if [ $# -gt 1 ]; then
    shift
    $0 $*
fi

exit 0

```

## Additional Sample Scripts

### JavaMonitor & wotaskd startup scripts for Linux:

<https://github.com/projectwonder/wonder/tree/integration/Utilities/Linux/StartupScripts>

### JavaMonitor & wotaskd startup scripts for Mac OS X (10.6 or later):

<http://wiki.wocommunity.org/display/documentation/Deploying+on+Mac+OS+X+10.6+%28Snow+Leopard%29>