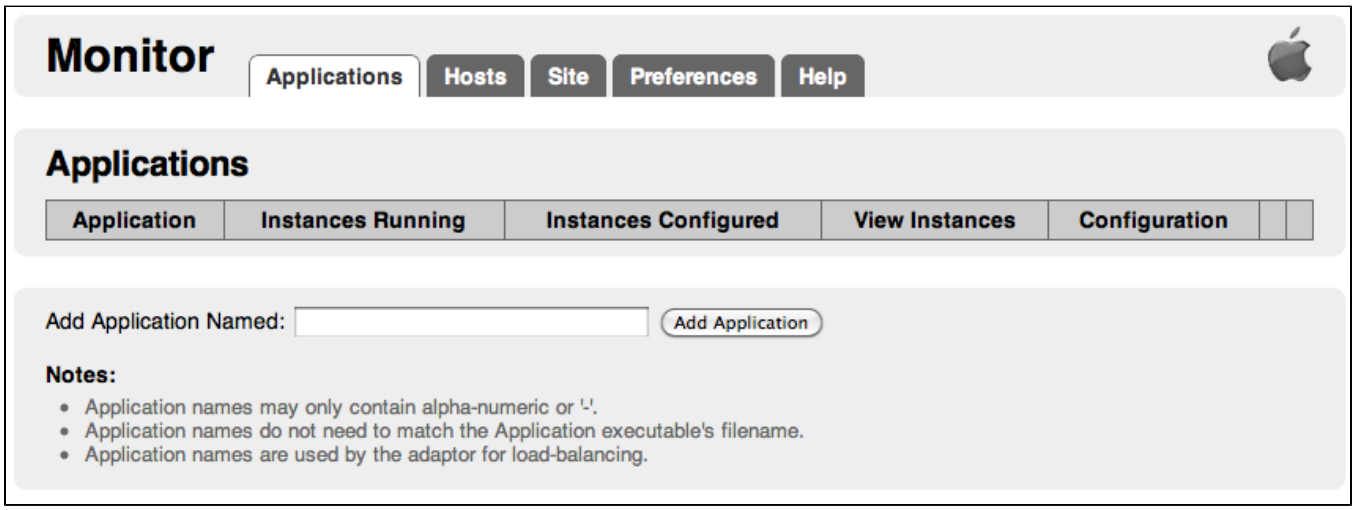
Wonder JavaMonitor and wotaskd



Introduction

JavaMonitor is a web-based tool used to configure and maintain deployed WebObjects applications. It is capable of handling multiple WebObjects applications, multiple instances of each application, and multiple WebObjects Applications Servers. In most cases you'll have one instance of JavaMonitor controlling all instances of your applications, even if they are spread across multiple Application Servers.

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

Apple released the original wotaskd and JavaMonitor to the community as open source in WebObjects 5.4. The source was then quickly modified and included in Project Wonder. Substantial improvements in both functionality and look-and-feel have been made over the years. These improved versions of **wotaskd.woa** and **JavaMonitor.woa** are available as standard Wonder applications.

Where To Get Them

You can either download them pre-built from Wonder's Jenkins build server or build them from the source code.

To build them from the Wonder source code, simply run the following command from the Wonder directory at the root of the Wonder source.

```
ant frameworks deployment.tools -Ddeployment.standalone=true
```

What this command does:

- **ant**: calls Apache Ant. It is assumed that you have this already installed.
- **frameworks**: tells Ant to build the "frameworks" target. This may not be needed if you already have Wonder built and installed in a location Ant can find automatically.
- **deployment.tools**: tells Ant to build the "deployment.tools" target. This is the target that builds both wotaskd.woa and JavaMonitor.woa. You absolutely need this one.
- **-Ddeployment.standalone=true**: argument will embed the required Wonder and WebObjects frameworks in built applications. You need this to ensure that the required frameworks are embedded in the built applications.

Bug Fixes to Apple's Versions

JavaMonitor

- Fixes an issue with the Application Delete page
- Selection in Application Detail page is now Ajax and is maintained

Improvements to Apple's versions

Automatic archive of SiteConfig.xml

On every change you make to an application's configuration, a backup of SiteConfig.xml will be created in, by default, /Library/WebObjects/Configuration.

Simplified/Automated Bouncing

In the "list instances" page, you get a "Bounce" action link. This action only work if you have at least one active instance and one inactive instance (only one inactive instance takes part in the bounce). What it does is :

- Find one inactive (i.e., not started) instance and start it
- Find the active instances (minus the one started in the previous step) and enable "Refuse New Session"
- Stop the active instances that are refusing new sessions when the minimum session count is reached
- Restart all but one of the instances that were just stopped and turn on "Auto-Recover"

This feature, from Pascal's understanding, allows you to upload new versions of your application, start up the new version and refuse sessions for the instances running on the older version. This is designed to work so that you always have just **one** inactive instance that is only used while the bounce is performed. While waiting for the existing sessions to end you will have only one instance of your app accepting new sessions; you need to determine if this is acceptable for your app or not.

Database Changes

It is uncertain what will happen if the new version of your application makes changes to your database schema (e.g., uses ERXMigrations). The old instances may raise exceptions before they can gracefully shut down because the schema no longer matches what the old application's EOF expects. **Try it out on a test server first and then update this page so everyone knows!**

Statistics

If you call `http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/wa/statistics`, JavaMonitor will send you back statistics, in serialized Property List format, about instances, per application.

Statistics Example Results

```
(
        {
                "configuredInstances" = "2";
                "maxSessions" = "0";
                "maxAvgIdleTime" = "2.078";
                "avgTransactions" = "44.0000";
                "sumSessions" = "0";
                "avgAvgTransactionTime" = "0.0985000";
                "refusingInstances" = "0";
                "avgSessions" = "0.0000";
                "maxTransactions" = "88";
                "applicationName" = "AjaxExample";
                "avgAvgIdleTime" = "1.0390000";
                "maxAvgTransactionTime" = "0.197";
                "runningInstances" = "2";
                "sumTransactions" = "88";
        },
        {
                "configuredInstances" = "2";
                "maxSessions" = "0";
                "maxAvgIdleTime" = "325.443";
                "avgTransactions" = "0.5000";
                "sumSessions" = "0";
                "avgAvgTransactionTime" = "0.00000";
                "refusingInstances" = "0";
                "avgSessions" = "0.0000";
                "maxTransactions" = "1";
                "applicationName" = "AjaxExample2";
                "avgAvgIdleTime" = "162.7215000";
                "maxAvgTransactionTime" = "0.0";
                "runningInstances" = "2";
                "sumTransactions" = "1";
        }
)
```

If JavaMonitor is configured with a password, and I hope you do, pass *pw=monitorpassword* as a argument to the query :

```
http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/wa/statistics?pw=monitorpassword

wget http://monitorhost:56789/cgi-bin/WebObjects/JavaMonitor.woa/admin/stop?
type=app&name=InstanceName&pw=yourPassword
```

Direct Actions in JavaMonitor for Management Tasks

You can do most of the standard management tasks you'd normally do in JavaMonitor's web UI by calling standard WebObjects Direct Actions. Instead of using the `/wa/` request handler though, these management tasks use a new `/admin/` request handler. These Direct Actions can be very useful, especially if you need to restart instances or other do tasks from the command line, from within Ant or other build or deployment systems.

List of available direct actions :

- **`info`** : Returns details (number of deaths, state, etc.), in JavaScript Object Notation (JSON) as specified
  `info?`**`type=all`**
  `info?`**`type=app&name=`**`AppName`
  `info?`**`type=ins&name=`**`AppName-InstanceNumber`
- **`running`** : Returns **`YES`** if **all** of the specified are running, `NO` if not
  `running?`**`type=all`**
  `running?`**`type=app&name=`**`AppName`
  `running?`**`type=ins&name=`**`AppName-InstanceNumber`
- **`stopped`** : Returns **`YES`** if **all** the specified is running, **`NO`** if not.
  `stopped?`**`type=all`**
  `stopped?`**`type=app&name=`**`AppName`
  `stopped?`**`type=ins&name=`**`AppName-InstanceNumber`

- **bounce** : Returns **OK** after Bouncing (see description above) as specified
  `bounce?type=all`
  `bounce?type=app&name=AppName`

- **clearDeaths** : Returns **OK** after clearing deaths as specified
  `clearDeaths?type=all`
  `clearDeaths?type=app&name=AppName`

- **turnScheduledOn** : Returns **OK** after turning on scheduling as specified. Call */turnScheduledOff* to do the opposite.
  `turnScheduledOn?type=all`
  `turnScheduledOn?type=app&name=AppName`
  `turnScheduledOn?type=ins&name=AppName-InstanceNumber`

- **turnRefuseNewSessionsOn** : Returns **OK** after turning on "Refuse new sessions" as specified
  `turnRefuseNewSessionsOn?type=all`
  `turnRefuseNewSessionsOn?type=app&name=AppName`
  `turnRefuseNewSessionsOn?type=ins&name=AppName-InstanceNumber`

- **turnRefuseNewSessionsOff** : Returns **OK** after turning off "Refuse new sessions" as specified
  `turnRefuseNewSessionsOff?type=all`
  `turnRefuseNewSessionsOff?type=app&name=AppName`
  `turnRefuseNewSessionsOff?type=ins&name=AppName-InstanceNumber`

- **turnAutoRecoverOn** : Returns **OK** after turning on "Auto Recover" as specified
  `turnAutoRecoverOn?type=all`
  `turnAutoRecoverOn?type=app&name=AppName`
  `turnAutoRecoverOn?type=ins&name=AppName-InstanceNumber`

- **turnAutoRecoverOff** : Returns **OK** after turning off "Auto Recover" as specified
  `turnAutoRecoverOn?type=all`
  `turnAutoRecoverOn?type=app&name=AppName`
  `turnAutoRecoverOn?type=ins&name=AppName-InstanceNumber`

- **forceQuit** : Returns **OK** after force quitting as specified. This could be useful to call from a monitoring system.
  `forceQuit?type=all`
  `forceQuit?type=app&name=AppName`
  `forceQuit?type=ins&name=AppName-InstanceNumber`

- **stop** : Returns **OK** after calling "Stop" as specified
  `stop?type=all`
  `stop?type=app&name=AppName`
  `stop?type=ins&name=AppName-InstanceNumber`

- **start** : Returns **OK** after calling "Start" as specified
  `start?type=all`
  `start?type=app&name=AppName`
  `start?type=ins&name=AppName-InstanceNumber`

To get details about all instances of all applications:
`http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/admin/info?type=all`

To get details about the **AjaxExample** application:
`http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/admin/info?type=app&name=AjaxExample`

To get details about instance **1** of the **AjaxExample** application:
`http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/admin/info?type=ins&name=AjaxExample-1`

Remote Control via REST Routes (for JavaMonitor)

If the control offered by the Direct Actions isn't enough, JavaMonitor allows additional control via REST calls. Between the two methods (Direct Actions, REST) you have almost full remote-control of JavaMonitor. Just make sure that your JavaMonitor installation is secure! Just like with Direct Actions, you need to append `?pw=XXXX` to the URLs if your JavaMonitor is password protected.

Examples of REST calls :

Adding a New Host

```
curl -X POST -d "{id: 'otherserver.com',type: 'MHost', osType: 'MACOSX',address: '192.168.20.5', name:
'otherserver.com'}" http://monitorhost:port/apps/WebObjects/JavaMonitor.woa/ra/mHosts.json
```

Fetching Details for All Applications

```
curl -X GET http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications.json
```

Adding a New Application

```
curl -X POST -d "{id: 'AjaxExample',type: 'MApplication', name: 'AjaxExample',unixOutputPath: '/opt/Local
/Library/WebObjects/Logs', unixPath: '/opt/Local/Library/WebObjects/Applications/AjaxExample.woa/AjaxExample'}"
http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications.json
```

Delete an Application

```
curl -X DELETE http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample.json
```

Adding a New Instance

```
curl -X GET http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample
/addInstance&host=localhost
```

Delete an Instance

```
curl -X GET http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample
/deleteInstance?id=1
```

Configuring the Site

```
curl -X PUT -d "{woAdaptor:'www.mydomain.com'}" http://monitorhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra
/mSiteConfig.json
```

Note that before configuring the site you must first add a host. If you attempt to configure the site prior to adding a host you will get an InvalidStateException.

Remote Control via REST Routes (for wotaskd)

Starting on August 10th 2012, wotaskd also have REST routes. By using them, you can do most of the tasks using REST and you might not even need JavaMonitor. Just like the JavaMonitor REST and Direct Actions, you need to append `?pw=XXXX` to the URLs if wotaskd is password protected.

Fetching Details for All Applications

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications.json
```

Adding a New Application

```
curl -X POST -d "{id: 'AjaxExample',type: 'MApplication', name: 'AjaxExample',unixOutputPath: '/opt/Local
/Library/WebObjects/Logs', unixPath: '/opt/Local/Library/WebObjects/Applications/AjaxExample.woa/AjaxExample'}"
http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications.json
```

Delete an Application

```
curl -X DELETE http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample.json
```

Adding a New Instance

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample
/addInstance&host=localhost
```

Delete an Instance

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample
/deleteInstance?id=1
```

Configuring the Site

```
curl -X PUT -d "{woAdaptor:'www.mydomain.com'}" http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra
/mSiteConfig.json
```

Starting all applications

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/start
```

Starting a specific application (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/start
```

Starting a specific instance of an application (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/start?id=1
```

Stopping all applications

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/stop
```

Stopping a specific application (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/stop
```

Stopping a specific instance of an application (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/stop?id=1
```

Force quit all applications

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/forceQuit
```

Force quit a specific application (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/forceQuit
```

Force quit a specific instance of an application (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/forceQuit?
id=1
```

Information about all applications

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/info
```

Information about a specific application (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/info
```

Information about a specific instance of an application (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/info?id=1
```

Check if all applications are running

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/isRunning
```

Check if a specific application is running (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/isRunning
```

Check if a specific instance of an application is running (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/isRunning?id=1
```

Check if all applications are stopped

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/isStopped
```

Check if a specific application is stopped (AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/isStopped
```

Check if a specific instance of an application is stopped (instance 1 of AjaxExample in this example)

```
curl -X GET http://wotaskdhost:port/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/AjaxExample/isStopped?id=1
```

Troubleshooting

If JavaMonitor won't start up check the troubleshooting deployment section. In particular pay attention to the WOTaskd Didn't Start Q&A.