Your First Deployment

Work in progress!

Introduction

Usually, people deploy Project Wonder applications in one of those two ways: using the "classic" deployment tools (wotaskd + Web server module) or using a servlet container (aka Tomcat/Jboss/etc.).

We will focus on the classic tools since this is the method that the majority of the community use. The classic deployment tools consists of:

- wotaskd. A daemon that acts as a watchdog to send lifebeat to the applications its manage + launching and stopping applications. Source code is available and it's actually a Project Wonder application!
- Module for your Web server software (Apache or IIS). If a module doesn't exist for your preferred Web server, you can use a CGI.
- JavaMonitor. A Web GUI to manage wotaskd configuration (which is a XML file). It's optional, and you can manage multiple instances of wotaskd with a single JavaMonitor installation. JavaMonitor also has REST APIs so that you can manage the configuration by command-line or by a JavaScript app.

But first, we need to actually make our projects ready for deployment.

Building the "products"

When you run your projects inside Eclipse, it uses the incremental builder and the "internal builds" are not ready for deployment (e.g. don't use the build from the "build" folder of the project, it's not a complete build!). You need to use Ant or Maven to actually build a full "product". Let's focus on Ant because, again, this is what the majority use. You can also use Jenkins, but Jenkins will need to call Maven or Ant to build the projects anyway.

One important thing to remember is that you need to build and install the frameworks before your build your applications. Building your applications will NOT add the frameworks to the product if you didn't build the frameworks first. So let's build and install the **BlogCommon** framework first.

In Eclipse, right-click on the **BlogCommon** project, and select **WOLips Ant Tools** -> **Install**.

By calling this, Eclipse will call Ant to build the framework and install it in (on OS X) **/Library/Frameworks**, so the framework should be at **/Library/Frameworks/BlogCommon.framework**.

You are now ready to build the applications. The procedure is the same as for frameworks, so right-click on the **BlogRest** project, and select **WOLips Ant Tools** -> **Install**. The application will install itself in (on OS X) **/Library/WebObjects/Applications/**. But there's a difference with a build for a framework: building an application will build three products:

- a .woa (in our case: **BlogRest.woa**) that is a regular application bundle.
- a tar+gzip archive of the application (in our case: **BlogRest-Application.tar.gz**)
- a tar+gzip archive of the Web server resources (the files that are in the **WebResources** folders of your projects). That file is name **<ApplicationName>-WebServerResources.tar.gz** (so in our case, **BlogRest-WebServerResources.tar.gz**).

The tar archive exists to make it faster to copy the application to your deployment server because it's two files instead of hundred and they are compressed.

Installing the deployment tools

We are now ready to install the deployment tools. The wiki has installation instructions for many platforms, so jump over to the instructions for your deployment platform. Running the deployment tools (at least wotaskd and the Apache module) is also useful on your development machine.

- Installing on Mac OS X
- Installing on Linux
- Installing on Windows
- Installing on FreeBSD

Configuring the applications with JavaMonitor

Before adding the application into JavaMonitor, you have to copy the applications on the deployment server. You can use any tools that you usually use to copy files on the remote server (scp, rsync, CyberDuck, etc.). Usually you need to copy the application to the following remote directory:

- /Library/WebObjects/Applications (on OS X)
- /opt/Local/Library/WebObjects/Applications (on UNIX systems other than OS X, like Linux, BSD or Solaris)

Copy the **BlogRest-Application.tar.gz** archive into the correct directory from the above list and uncompressed it (*tar zxf BlogRest-Application.tar.gz*). This will create a new directory named **BlogRest.woa**.

One last step to do for the application. Usually, the application will run with the *appserver* user and the *appserveradm* group. You need to make sure that either the group or the user can run the application. The best way to do this is to change the group of the files to be *appserveradm*. Your user on the deployment server should also be part of that group. To change the group:

```
chgrp -R appserveradm BlogRest.woa
```

You also need to install the Web server resources to a different location. Why is that? Because it's better to serve static resources like images and CSS from the Web server software (Apache, IIS, etc.) than from the Web application to get better performance and caching.

Usually, the Web server resources goes to:

- /Library/WebServer/Documents/WebObjects (on OS X)
- /opt/Local/Library/WebServer/Documents/WebObjects (on UNIX systems other than OS X, like Linux, BSD or Solaris)

Copy the **BlogRest-WebServerResources.tar.gz** archive into the correct directory from the above list and uncompressed it (*tar zxf BlogRest-WebServerResources.tar.gz*). Doing this command will result in a new directory called **BlogRest.woa**.

Everything on the file system is ok, the next step is to add the application to JavaMonitor. If the development tools have been installed correctly, JavaMonitor will run on port 56789 on your deployment server. For example, if the DNS name of your deployment server is my.host.com, the URL will be http://my.host.com:56789

After the **Add Application Named** label, enter **BlogRest** into the text field and click **Add Application**.

You will get to a page where you need to specific the path to the launch script inside the application bundle. If the development server is on OS X, the path will be:

```
/Library/WebObjects/Applications/BlogRest.woa/BlogRest
```

Put that path in the text field next to the **MacOSX** label.

On other UNIXs, the path is:

```
/opt/Local/Library/WebObjects/Applications/BlogRest.woa/BlogRest
```

Put that path in the text field next to the **Unix** label.

It would also be useful to log problems to a log file. For now, we will put the log into */tmp*. That means that the log will be lost when you reboot the server or if you have tools that clean the */tmp* directory every day, but for now it would do the job.

To specify the path to the log file, go to the **Output Path** section and enter */tmp* in the text field next to **MacOSX** (if your deployment box is on OS X) or next to **Unix** (if your development is UNIX but not OS X).

The next step is to click on the **Push All** button, which saves the application configuration. Now, we can add instances of the application. To do so, click on the **Detail View** link.

You will see a section saying _Add: 1 instance(s) on host: _. Click the **Add** button. That will add one instance of the application. It's possible to run multiple instances of an application, mainly to load balance requests and also to have other working instances in case an instance is going AWOL. But for now, running one instance is enough.

If the application launches successfully, a link on the application name will appear after 30 seconds in JavaMonitor. If a link does not appear after a minute and the status is still **STOP**, something is wrong. Check the log that was created in /tmp (the log should be /tmp/WOCommunity-1) to check for any errors.

If the log file does not exist, 99% of the time, the problem is file permissions. Make sure that the launch script for the application (BlogRest.woa/BlogRest) is accessible for the **appserver** user. If the file permissions are ok, you can launch the script manually with the **appserver** user by doing:

On OS X:

```
sudo su - appserver
cd /Library/WebObjects/Applications/BlogRest.woa/
./BlogRest
```

On other UNIXs:

```
sudo su - appserver
cd /opt/Local/Library/WebObjects/Applications/BlogRest.woa/
./BlogRest
```

If the application can start from the command line, it should start by the development tools too. If you see errors when launching the application by command line, fix them and try to launch it by command line again.