Alternative Ant Build Script for Fully Embedded and Split Install Bundles

> This script can be used for WOLips 3.3.x only, the build.xml in WOLips 3.4.x and later already have the split install/embedded stuff built-in.

WOLips 3.4.x, and later, Side-note

> If you are updating an older project, make sure you get the latest build.xml file. Create a new project, then copy/paste the contents of the fresh build.xml file into your project's build.xml file.

For WOLips 3.4.x and later, while embedding is built-in, it's not enabled by default. To enable embedding:

- make sure you are in the WO Explorer view
- right-click your project folder, select Properties, select WOLips Deployment
- check the related, if not all, options under Embed Frameworks

To create a versioned/dated bundle of your app and resources:

- make sure you are in the Navigator view
- edit build.properties, and add:
- `build.app.name=MyApp-2009-07-14`

There is a known bug with WO 5.4.x (for those not using the latest Wonder release) regarding proper linking to your web server resources within the embedded frameworks. The WOFrameworksBaseURL isn't set correctly. To do this you'll need to programmatically set this within your Application constructor:

- `setFrameworksBaseURL("/WebObjects/MyApp-2009-07-14.woa/Frameworks");`

Within build.properties (I may need to be corrected on this), the best approach to linking your embedded framework's web server resources automatically is to include (however the bug noted above breaks this):

- `frameworksBaseURL=/WebObjects/${build.app.name}.woa/Frameworks`

Introduction

The previous tutorial articles provide details of customization of the default ant build script that is present in your WebObjects projects. This article provides a fully custom build script that you can just drop into your project and use it as an alternative to the default build.xml. This script is called the "split_embedded_build" script. Along with providing the script, this article explains in detail what deployment artifacts are produced by the build. Hopefully the functionality in this script will make their way into the standard script someday with configuration parameters settable in a simple way in the build.properties file.

Pre-requisites

This script only works with the modern WOLips project format and the WOnderApplication project template format (since the layout of regular projects is now similar to Wonder layout). However the script is easily changed to accommodate the "old" standard format by changing the dir names in the script (for example "Sources" in the new project layout was "src" in the old layout)

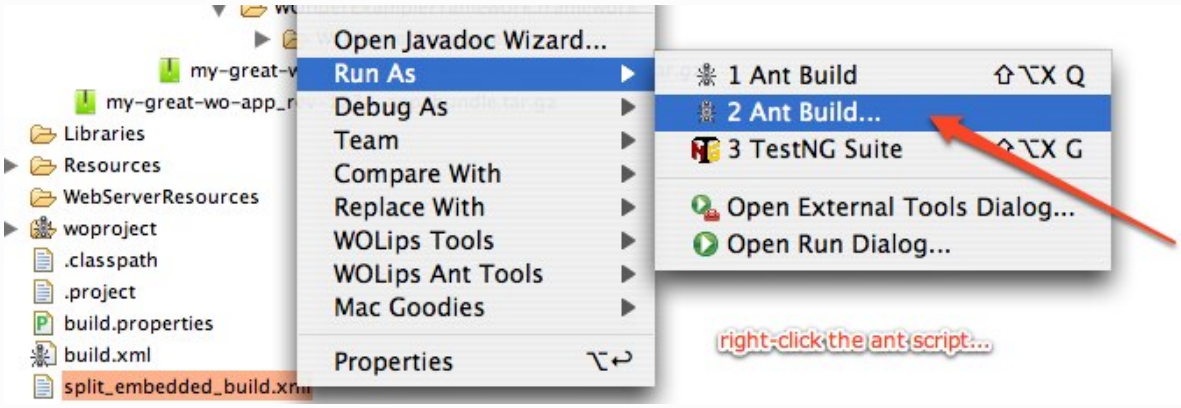
> **Modern WebObjects Project Layout**
>
> This layout is the standard layout for all WebObjects projects created by WOLips (rev #4735 at least). The distinct folder names indicate if your project is in this modern format and if yours is the same, then this ant script should work.

Installation and Usage of the Script

Have no fear , you will not break anything by dropping this script as is into your project and trying it out. You can just use it as is without replacing your default build.xml

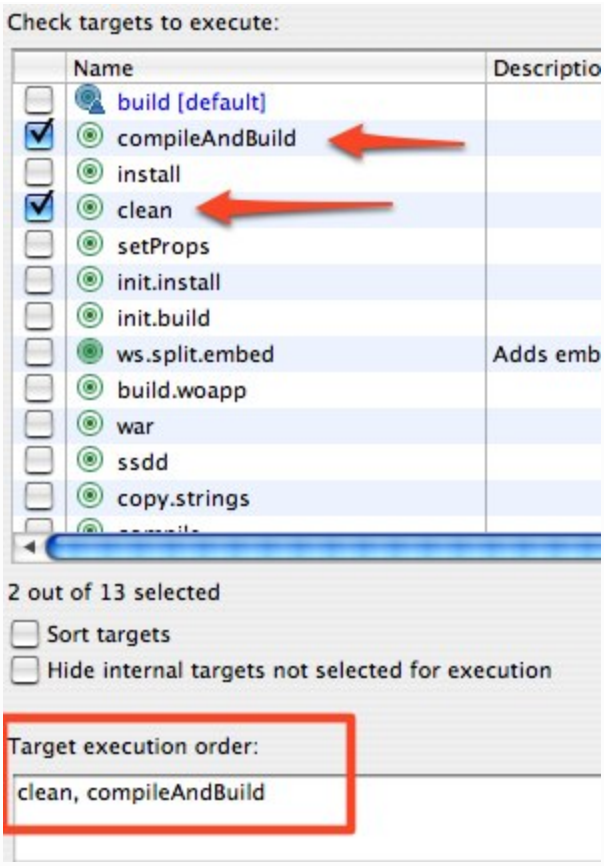
- So, download the script by right-clicking this link and selecing Save As... or whatever your browser uses for saving contents at a link destination.
- Drag it to the root of your project.
- Open the Ant run configuration for the script by right-clicking on the script and selecting the menu shown below.

Using the Ant Script



- Next select the "clean" and compileAndBuild" tasks. If necessary use the "Order..." button to ensure that "clean" comes before "compileAndBuild".

Configure the Ant Tasks to be run



- Click the Run button to run the ant script and your fully embedded split install deployment bundles will be created in the "dist" folder.

Customizing Application/Bundle Name

It is common to want to use custom bundle build names. There are a number of advantages to having a unique application name for each build:

- For example, you might want to have a name that reflects the current revision so that a clean bundle is deployed without overwriting older version bundles on the servers.
- If the current and new versions of the app are compatible with the current database schema, smooth zero-downtime "rolling upgrades" are possible by using this strategy... and the ability to switch back to the older version if an unexpected deployment problem occurs
- Because we are using fully embedded split bundles, the unique deployment name will ensure that web server resources all get a new URL ensuring that older versions cached by browsers will not be used for new deployed version apps ... which can be important for javascript and image updates for example.

OK, so getting to the point, an optional custom app name can be set in the project's build.properties file by simply adding a property "build.app.name" as shown below.

Customizing the Application Build Name

build.properties

```
classes.dir=bin
project.name=WOnderExampleApp
project.name.lowercase=wonderexampleapp
principalClass=wonder.example.app.Application
customInfoPListContent=
webXML=false
webXML_CustomContent=


## Just uncomment this and set the property value to have a custom application
# name, bundle name, executable name, etc.
# This propert is read by the custom split_embedded_build.xml ant build file
build.app.name=my-great-wo-app_rev-1234
```

Features

The desirable features of this script are

- Fully embedded - your deployed bundle reflects the frameworks you developed and tested with instead of risky assumption that the server has the exact same WebObjects and 3rd party frameworks configuration and versions that you had on your development machine.
- Custom application/bundle names - new version has new URLs for webserver resources that are often cached by browser so the new version will be requested since the URL is different.
- Self-contained deployment bundles - all required frameworks are embedded - minimal dependency on target server configuration and versions. For example, there is no need to install Wonder frameworks on your deployment server since the ones you developed and tested with are embedded in the deployment bundles. Experience has shown that this improves consistency of deployment reliability.
- Compressed gzipped tar archives of both application and webserver bundles are created, ready for copying to target server(s)
    - This is easily decompressed on the server using
    - $ sudo tar -xvzf bundle.tar.gz

Anatomy of Embedded Split Install Bundles produced by this Ant Script

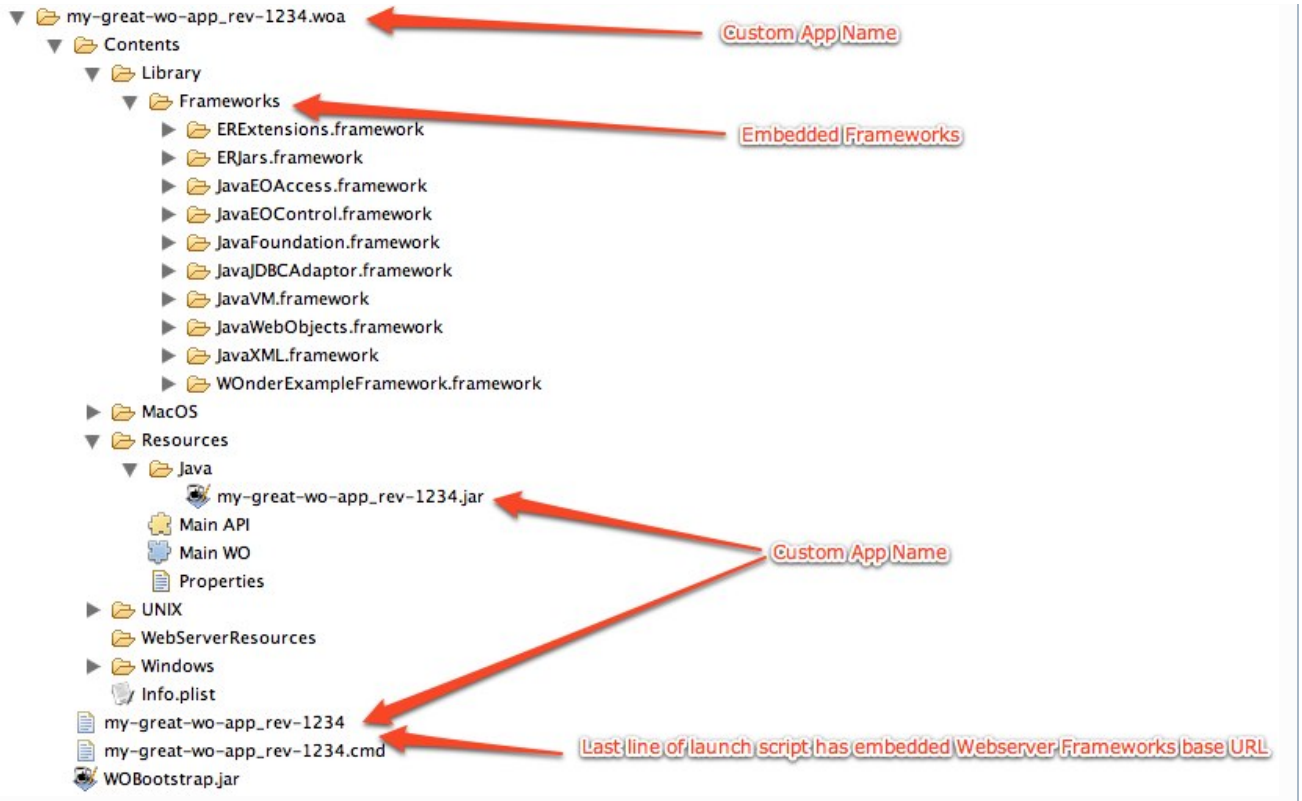
The Deployment Files

The two compressed "ready-to-deploy" bundles are highlighted below

The notable aspects of the embedded build structure are shown below.

Application Bundle

Embedded Application Bundle Layout



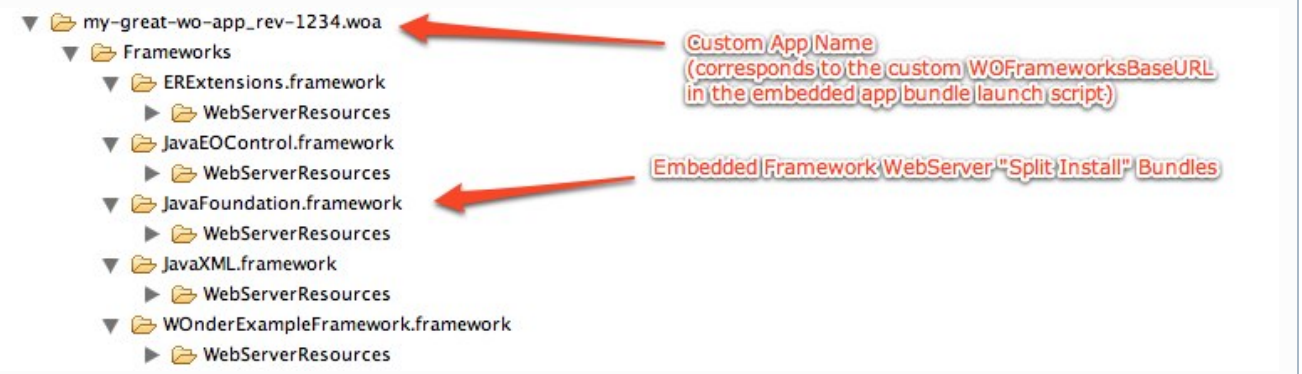
- Opening the app executable (command line script at bundle root having the application name as its name) and looking at the end will reveal that the WOFrameworksBaseURL references the frameworks inside the webserver embedded bundle.
- Examining the Contents/MacOS/MacOSClassPath.txt file will reveal that the java classpath refers to all the jars inside the embedded bundle.

The application bundle typically gets installed as follows:

- Install Location = /Library/WebObjects/Applications/
- chown (ownership) = appserver:appserveradm
- chmod (permissions) = 755 (750 if you wish, but your server login id needs to be a member of appserveradm group to cd into the bundle)

WebServer Bundle

Embedded WebServer Bundle Layout



The webserver bundle typically gets installed as follows:

- Install Location = /Library/WebServer/Documents/WebObjects/
- chown (ownership) = root:wheel
- chmod (permissions) = 755

Compatibility

This script was developed and tested on Mac OS X 10.4.10, Eclipse 3.3.1 and WOLips #4735. It should be compatible with later versions of WOLips and Eclipse. It works with regular WebObjects apps and Wonder based apps. I don't have a Windows Eclipse/WOLips installation, so if you are a Windows user, please feel free to make necessary changes and submit revised build script if this does not run on Windows.