Embedding WOFrameworks

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");`

Another way to fix this bug is to set the following launch parameter:

-DWOFrameworksBaseURL=/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`

There was an old bug in the build.xml file but fixed if you have WOLips more recent than october 2010:
http://issues.objectstyle.org/jira/browse/WOL-979

Introduction

Below is outlined "simple embedding" concepts, however full embedding **and** split-installing are really recommended. This provides fully versioned self-contained bundles of both the application deployment bundle and the webserver deployment bundle. Read the docs on that technique for more **advantages** . This technique fully explained along with a ready-to-use ant build script is available at:

Alternative Ant Build Script for Fully Embedded and Split Install Bundles

Simple Approach

This simple approach can be used to embed frameworks already **installed** on your development machine right into the app.woa bundle.

Open your build.xml project file and change embed="false" to embed="true" for the appropriate directory paths in this section of the build.xml file inside the woapplication task

Default build.xml snippet

```
<frameworks root="${wo.wolocalroot}" embed="false">
        <patternset>
                <includesfile name="woproject/ant.frameworks.wo.wolocalroot" />
        </patternset>
</frameworks>
<frameworks root="${user.home}" embed="false">
        <patternset>
                <includesfile name="woproject/ant.frameworks.user.home" />
        </patternset>
</frameworks>
<frameworks root="${wo.wosystemroot}" embed="false">
        <patternset>
                <includesfile name="woproject/ant.frameworks.wo.wosystemroot" />
        </patternset>
</frameworks>
```

The most common choice is to embed locally installed WOnder and 3rd party frameworks that your app references by setting embed="true" on the wolocalroot frameworks dir as shown below:

Changes to build.xml to embed frameworks installed in /Library/Frameworks dir

```
<frameworks root="${wo.wolocalroot}" embed="true">
        <patternset>
                <includesfile name="woproject/ant.frameworks.wo.wolocalroot" />
        </patternset>
</frameworks>
<frameworks root="${user.home}" embed="false">
        <patternset>
                <includesfile name="woproject/ant.frameworks.user.home" />
        </patternset>
</frameworks>
<frameworks root="${wo.wosystemroot}" embed="false">
        <patternset>
                <includesfile name="woproject/ant.frameworks.wo.wosystemroot" />
        </patternset>
</frameworks>
```

The various dirs referenced in the build.xml from which you can select the embed option are:

dir name	Description
wo.wolocalroot	/Library/Frameworks/ Embedding these is most useful especially if you the codebase for these, like WOnder frameworks or your own, is frequently updated
user.home	~/Library/Frameworks/
wo.systemroot	/System/Library/Frameworks/ This is where Apple's WebObjects frameworks are installed and if your WebObjects version is consistent between your deployment and development platforms, then embedding these is not of much benefit.

Remember to...

If you are working with framework source in your Eclipse workspace (which you should be!), then it is critical that you build and install all frameworks before building the deployment bundles to ensure that the embedded frameworks are the same as the source you have been developing and testing with. See next subsection for one approcah to automating this.

Pre-installing Workspace Frameworks Before Embedded Build

You could write a task to install the frameworks directly from the workspace, for example:

```
<!-- Install my own dependent frameworks in final local install location before embedding -->
        <target name="installMyFrameworks">
                <ant dir="../WKDemography" target="install" inheritall="false" />
                <ant dir="../WKEmailData" target="install" inheritall="false" />
                <ant dir="../WKEOFExtensions" target="install" inheritall="false" />
                <ant dir="../WKFoundation" target="install" inheritall="false" />
                <ant dir="../WKPrototypes" target="install" inheritall="false" />
                <ant dir="../WKRemoteClient" target="install" inheritall="false" />
                <ant dir="../WKReports" target="install" inheritall="false" />
                <ant dir="../WKWebObjects" target="install" inheritall="false" />
        </target>
```

... and then you could put a depends attribute on the build target to ensure the frameworks are installed before the embedded build task is executed:

```
<target name="build.woapp" depends="installMyFrameworks">
```

Advanced Approach

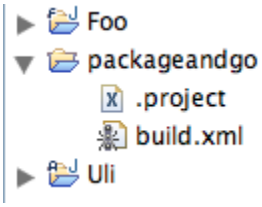
Caveat: This example assumes that each framework you want to embed has an ant "build.xml" file with a "compileAndBuild" target. For frameworks you create yourself using WOLips, you will have this, but for a case where you are checking out individual Wonder framework projects from CVS, then you probably will not have such a build.xml file, so this approach is not quick to implement in that case unless you want to spend a lot of time working on ant build files to resolve the missing build.xml having a compileAndBuild target for those external projects. (KK 3/30/2007)

See also FrameworkSet documentation.

Example for embedding WOFrameworks.

It's often a good idea to create your own targets in the build.xml or even create your own build.xml (with a different name). This leaves the door open for an update of the default build.xml.

Assume two projects: One named Foo(a framework) and the other named Uli (an application). The parent folder has another folder named packandgo.



The packageandgobuild.xml from the application. Two minor changes to the default build.xml:
1 application target

```
<!-- package and go example-->
<frameworks root="../packageandgo/frameworks" embed="true">
    <include name="*.framework"/>
</frameworks>
```

2 compile target

```
<!-- package and go example-->
<fileset dir="../packageandgo/frameworks">
    <include name = "**/*.jar"/>
</fileset>
```

The build.xml from the packageandgo folder:

Just invoke ant in the packageandgo folder and grap the App from the applications folder within the packageandgo folder.