WOApplication

WOApplication Task Manual

This has been transcribed by Andrew Lindesay from the old WOProject site.

Description

`WOApplication` is an Ant task to build WebObjects applications from a set of files. It does not enforce any particular project structure and can be used to create applications without using the ProjectBuilder or XCode.

Properties

WOApplication behavior depends on various properties in the wobuild.properties file that specify common locations of the frameworks used by the application. These properties can be initialized with the ant script woproperties.xml. Per default the woapplication task expects this file in $user.home /Library/wobuild.properties. If no file is found the woapplication task resolves the property WOBUILD_PROPERTIES from the java properties and the environment(in this order) to find the file.

Property from the wobuild. properties file	Description	Default
wo.wosystemroot	Usually this is a WebObjects installation directory, like "C:\Apple".	NEXT_ROOT environment variable, or root directory "/".
wo.wolocalroot	Usually this is a "Local" directory under the directory specified by "wo. woroot".	${wo.woroot}/Local
wo.homeroot	Usually this is a user home directory.	${user.home}

Parameters

Attribute	Description	Required
name	Name of the application (without .woa extension).	Yes
chmod	Optional value for the chmod command executed by the WOApplication task. The default is "gu+x". The chmod command is only executed on the first build of the application and only on "Unix" platforms. The default is "750".	No
destDir	Destination directory where the application woa should be created.	Yes
wsDestDir	Destination directory where WebServerResorces should be copied during split install (presense of this parameter will trigger split install). WebServerResources will be created under wsDestDir/WebObjects/AppName.woa/Contents/.	No
principalClass	Subclass of WOApplication to use.	Application
customInfoPListContent	String to append to the Info.plist.	No
stdFrameworks	If set to true, a set of standard frameworks will be associated with the deployed application (default is true). "Standard" frameworks are: JavaWebObjects, JavaWOExtensions, JavaEOAccess, JavaEOControl, JavaFoundation, JavaJDBCAdaptor, JavaXML.	No
jvm	Path to the JVM binary. Defaults to 'java'	No
jvmOptions	String for the JVM options in the classpath.	No
jdb	Path to the JVM binary. Defaults to 'java'	No
jdbOptions	String for the jdb options in the classpath.	No
webXML	Generate web.xml	No
webXML_WOROOT	Optional parameter for the web.xml	No
webXML_LOCALROOT	Optional parameter for the web.xml	No
webXML_WOAINSTALLROOT	Optional parameter for the web.xml	No
webXML_WOAppMode	Optional parameter for the web.xml	No
webXML_WOtaglib	Optional parameter for the web.xml	No

webXML_CustomContent	Optional parameter for the web.xml	No
cfBundleVersion	CFBundleVersion for the Info.plist. Default is no version.	No
cfBundleShortVersion	CFBundleShortVersionString for the Info.plist. Default is no version.	No
cfBundleID	CFBundleIndetifier for the Info.plist. Defaults to 'com.myapp'.	No
javaVersion	JVMVersion for the Info.plist. Used for selecting JVM compatibility. Defaults to '1.5+'.	No
frameworksBaseURL	This denotes the directory containing the framework webserver resources bundles (MyFramework.framework, ERExtensions.framework, etc.). This is typically useful for a embedded split install whereby all webserver resource frameworks (all = localroot and systemroot) are embedded in the webserver split install application bundle. The advantage of doing this is of course is clean upgrades with current versions of framework resources being used independently of what is installed on the deployment server. Defaults to '/WebObjects/Frameworks'. The URL is relative to the webserver document root. For example: frameworksBaseURL="/WebObjects/${project.name}.woa/Frameworks"	No
startupScriptName	.....	No

Nested Elements

classes

The nested `classes` element specifies a FileSet. All files included in this fileset will end up in the `Contents/Resources/Java/*.jar` file of the application.

resources

The nested `resources` element specifies a FileSet. All files included in this fileset will end up in the `Contents/Resources` directory of the application. For the discussion of resource localization issues follow this link.

wsresources

The nested `wsresources` element specifies a FileSet. All files included in this fileset will end up in the `Contents/WebServerResources` directory of the application. For the discussion of resource localization issues follow this link.

frameworks

The nested `frameworks` is a FrameworkSet structure that specifies the names of the WebObjects Frameworks that this application is dependant upon. The jar files from these frameworks will be referenced in varios platform-specific depoyment files (such as CLASSPATH.TXT), and specified in the web.xml classpath if an application is deployed as a servlet. When building a FileSet, path should match up to the "*.framework" directory (no need to match individual JAR files).

To embed frameworks set embed to true.

Nov13/08 email from Kieran Kelleher

Q: *Kieran Kelleher*
With regards the woapplication task and nested elements, what does the eclipse="true" attribute of the 'frameworks' nested element do?
```
<frameworks dir="ProjectLocal" embed="${embed.ProjectLocal}" eclipse="true" />
```

A: *Mike Schrag*
eclipse="true" makes the <frameworks> tag parse your .classpath file to determine framework dependencies.

lib

The nested lib element specifies a FileSet. This should be a fileset of jar libraries required by your application. All files in this fileset will end up in the `Resources/Java` folder of the application, and will be included on the classpath for this application.

To embed jars set embed to true.

otherclasspath

This nested element is used to add an arbitrary directory or file path to the runtime classpath. For example on Mac OS X, the directory at APPROOT /Resources/SSLKeyStore/ can be added to the runtime classpath by doing something like this:

```
<woapplication>
   [...]
   <otherclasspath dir="." embed="true">
       <include name="Resources/SSLKeyStore" />
   </otherclasspath>
</woapplication>
```

Examples

Create an application "MyApp" with a set of standard frameworks:

```
<taskdef name="woapplication" classname="org.objectstyle.woproject.ant.WOApplication">
     <classpath refid="classpath"/>
</taskdef>

<woapplication name="MyApp" destDir="${dist}/WebObjects/Applications">
   <classes dir="${build}/common">
      <exclude name="*.properties"/>
   </classes>
   <classes dir="${build}/business">
       <exclude name="*.properties"/>
   </classes>
   <resources dir="src/resources">
       <include name="*.eomodeld/**"/>
       <include name="*.wo/**"/>
   </resources>
   <wsresources dir="src/frameworks/WSResources">
       <include name="Images/**"/>
   </wsresources>
</woapplication>
```

Create an application "MyApp" with no implicit standard frameworks and a set of custom frameworks located under HOMEROOT (defined from the value of `wo.homeroot` property):

```
<woapplication name="MyApp" stdFrameworks="false" destDir="${dist}/WebObjects/Applications">
   ...
   <frameworks dir="../Frameworks">
      <include name="**/*.framework"/>
   </frameworks>
</woapplication>
```

Create an application "MyApp" with embedded frameworks:

```
<woapplication name="MyApp" destDir="${dist}/WebObjects/Applications">
   ...
   <frameworks dir="../Frameworks"embed="true">
      <include name="**/*.framework"/>
   </frameworks>
</woapplication>
```