Большое пособие по Apache Ant

|  |  |
| --- | --- |
| категория [Java Tools](http://www.quizful.net/category/java_tools) | |
| дата | 16.07.2009 |
| автор | [ieronim](http://www.quizful.net/user/ieronim) |

[Disclaimer: Данная статья была переведена в рамках "Конкурса на лучший перевод статьи" на сервисе Quizful. Ссылка на оригинал находится внизу страницы.]

**Apache Ant** - мощный инструмент для преобразования ваших структур разработки в структуры развертывания приложений. Он является декларативным и все инструкции командной строки, используемые для развертывания приложения, представляются простыми **XML**-элементами. Не вдаваясь в детали, данное пособие шаг за шагом расскажет вам как организовать сборку web-приложения с помощью одного-единственного **XML-файла**.

Моя структура разработки состоит из следующих каталогов:  
  
   1. **web** - каталог для всех JSP, HTML, JavaScript-ов и таблиц стилей. Вы можете создать подкаталоги, если требуется, для каждого набора файлов;  
   2. **src** - каталог для файлов java-классов, содержащая POJO или сервлеты;  
   3. **etc** - каталог для всех конфигурационных файлов, вроде наиболее распространенного web.xml;  
   4. **lib** - каталог для jar-файлов, необходимых для работы моего приложения. Я даже включил servlet-api.jar в этот каталог, так как вам, возможно, придется разворачивать ваше web-приложение на удаленном сервере после компиляции;  
   5. **build** - временный каталог для хранения скомпилированных файлов;  
   6. **dist** - каталог, куда будет помещен окончательно упакованный war-файл (дистрибутив);  
   7. **ant** - каталог для файла build.xml и внешних properties-файлов.  
  
Все эти каталоги находятся в родительской директории “WebApp“.  
  
Вот три вещи, которые вам надо держать в уме для создания файла сборки Ant:  
  
   1. **tasks (задачи)** - соответствуют инструкциям командной строки, например javac, war и т.д. Группа задач может выполняться последовательно с помощью указания целей.  
   2. **target (цель)** - это что-то наподобие функции, в которой вы размещаете задачи, таким образом получая возможность повторно вызывать их без дублирования.  
   3. **properties (свойства)** - используются для определения переменных в ваших файлах сборки, что очень полезно, в случае когда название проекта или названия директорий могут изменяться.  
  
Одна из самых приятных особенностей Ant в том, что вы можете хранить файл свойств отдельно, вместо того, чтобы определять все переменные внутри файла сборки. Файлы свойств состоят из перечисления всех требуемых переменных и их значений в виде пар имя-значение и представляют собой обычный текстовый файл. В данном пособии я буду использовать внешний файл свойств.  
  
При написании пособия, я использовал следующие инструменты:  
  
   1. Java версии 1.5  
   2. Apache Tomcat 5.0  
   3. Apache Ant 1.6.5  
  
Итак, мы готовы приступить к освоению Ant. Для этого создайте файл с именем “build.properties“. Это будет наш внешний файл с переменными:  
  
root.dir=..  
lib.dir=lib  
src.dir=com  
conf.dir=etc  
web.content=web  
project.name=WebApp  
build.dir=build  
  
**ВНИМАНИЕ:** обязательно нужно нажать ENTER, чтобы начать новую строку после последней, иначе Ant выдаст ошибку “BUILD FAILED for some unknown reason”  
  
Нет необходимости давать переменным имена с точкой. Вы можете использовать такие имена как “projectName”, но значение не должно содержать кавычек. То есть НЕ должно быть чего-то вроде projectName=”WebApp”  
  
Для создания файла сборки, мы должны помнить, какие действия необходимы для развертывания проекта. Чтобы упростить себе жизнь, просто создайте модули (targets), для всех действий при развертывании web-приложения:  
  
   1. **clean**- удалить результаты предыдущих развертываний приложения, если они были  
   2. **init** - создать необходимую структуру для развертывания  
   3. **compile** - скомпилировать ваши сервлеты или POJO  
   4. **copy** - скопировать скомпилированные файлы и web-содержимое в структуру развертывания, созданную модулем init  
   5. **war** - создать war-файл и открыть браузер  
  
Начните писать файл **build.xml** в каталоге ant следующим образом:

<project name="AppBuilder" default="war" basedir="..">  
    <property file="ant/build.properties"/>  
</project>

Теперь мы должны установить classpath для servlet-api.jar для компиляции наших сервлетов, поэтому поместите servlet-api.jar в каталог lib вашей структуры разработки. Проверьте свойство “default” у элемента <project>. В нем установлен ПОСЛЕДНИЙ модуль (target) файла сборки. В нашем случае это “war”  
  
Все последующие XML-элементы будут размещены внутри **<project></project>**, созданного ранее. Напишите элемент для задания classpath,

<path id="classpath">  
     <fileset dir="${lib.dir}" includes="servlet-api.jar"/>  
</path>

Теперь вам надо прописать все необходимые jar-ы в classpath для успешной компиляции исходных файлов java. ${lib.dir} используется для получения значения переменной “lib.dir”, т.е. lib.  
  
Теперь пройдемся по модулям (targets) упомянутым в списке выше:  
  
1. **clean**  
  
<target name="clean">  
    <echo>Cleaning the ${build.dir}</echo>  
    <delete dir="${build.dir}"/>  
</target>  
  
Здесь я удаляю каталог сборки на случай если он уже существовал с ранее скомпилированными файлами  
Элемент <echo> служит только для вывода текущего действия в командной строке.  
  
2. **init**

<target name="init" depends="clean">  
    <echo>Creating the build directory</echo>  
    <mkdir dir="${build.dir}/WEB-INF/classes"/>  
    <mkdir dir="${build.dir}/WEB-INF/lib"/>  
</target>

Здесь я создаю обычную структуру развертывания, требуемую для tomcat, а именно WebApp/WEB-INF/classes, и т.д. Не имеет значения, существовала ли директория WEB-INF до создания каталога classes. Ant автоматически создает все необходимые родительские каталоги, если их не было.  
  
3. **compile**

<target name="compile" depends="init">  
    <echo>Compile the source files</echo>  
    <javac srcdir="${src.dir}" destdir="${build.dir}/WEB-INF/classes">  
        <classpath refid="classpath"/>  
    </javac>  
</target>

Наиболее важный и наиболее богатый на ошибки шаг. Убедитесь, что classpath настроен корректно в элементе <path>. Если все сделано правильно, то все файлы в директории “src.dir” успешно скомпилируются и переместятся в build\WebApp\WEB-INF\classes. Обратите внимание на свойство “depends” элемента <target>. Оно используется для последовательной связи модулей (targets).  
  
4. **copy**

<target name="copy" depends="compile">  
    <copy todir="${build.dir}/WEB-INF">  
        <fileset dir="${conf.dir}"/>  
    </copy>  
    <copy todir="${build.dir}">  
        <fileset dir="${web.content}"/>  
    </copy>  
    <copy todir="${build.dir}/WEB-INF/lib">  
        <fileset dir="${lib.dir}"/>  
    </copy>  
</target>

Здесь я просто копирую скомпилированные классы и web-содержимое в соответствующие каталоги структуры развертывания.  
  
5. **war**

<target name="war" depends="copy">  
    <echo>Building the war file</echo>  
    <war destfile="${dist.dir}/${project.name}.war" webxml="${build.dir}/WEB-INF/web.xml">  
        <fileset dir="${build.dir}"/>  
    </war>  
</target>

Это последний модуль (target) в моем файле build.xml, создающий WebApp.war, необходимый для развертывания. “war” - инструкция Ant которой вы передаете путь к web.xml и каталог содержащий структуру развертывания, то есть, в нашем случае, директорию “build”. destfile - окончательное расположение и имя war-файла, которое должно принять вид dist\WebApp.war после отработки скрипта.

Запуск скрипта

Сохраните build.properties и build.xml в каталоге ant. Убедитесь, что в переменной среде ANT\_HOME прописан каталог bin для вашей установки ant. Теперь, все, что вам нужно сделать - запустить команду ant на файле сборки таким образом:  
  
C:\> cd WebAppant\ant  
C:\WebAppant\ant> ant

Вот и все. В случае успеха, ваши разработки в итоге станут полностью упакованным WebApp.war.

Заключительные слова

Итак, из наших исходных файлов, с помощью одного XML-файла и замечательного инструмента Ant, мы сделали полностью упакованный war-файл, готовый к развертыванию в tomcat manager. Можно копать глубже для создания сложного build.xml, использующего полный набор элементов, предоставляемый Ant-ом. Но цель данного пособия лишь ознакомить вас с использованием Apache Ant. Если у вас возникли какие-либо трудности в понимании конкретных элементов, пожалуйста, прочтите их подробное описание в документации Apache Ant.