Table des matières

ant_introduction

Plus d'informations pour ANT :

http://ant.apache.org/manual/index.html

http://www.jmdoudoux.fr/java/dej/chap-ant.htm

https://tomcat.apache.org/tomcat-7.0-doc/appdev/build.xml.txt

http://www.tutorialspoint.com/ant/

http://www.dzone.com/tutorials/java/ant/ant-sample-build-file-war-1.html

Introduction

ANT est un outil Java (nécessite d'avoir le jdk installé) qui permet d'automatiser les tâches répétitives telles que la compilation, la création d'archives, la génération de la documentation des classes, l'exécution des tests unitaires, l'envoi de données sur un serveur, ...

Pour l'installer (http://ant.apache.org/manual/install.htm) : le dézipper, créer la variable d'environnement `ANT_HOME`, vérifier `JAVA_HOME`, ajouter `%ANT_HOME%/bin` à la variable `PATH`.

Il fonctionne grâce à un fichier au format XML nommé `build.xml`.

On y définit un projet, des variables (optionnel) et des tâches (ou **targets**).

Le texte ci-dessous est extrait du site http://www.tutorialspoint.com/

Vous trouverez sur cette page l'exemple de builx.xml du site Tomcat.

Ant - Creating WAR files

http://www.tutorialspoint.com/ant/ant_creating_war_files.htm

Creating WAR files with Ant is extremely simple, and very similar to the creating JAR files task. After all, WAR file, like JAR file is just another ZIP file.

The WAR task is an extension to the JAR task, but it has some nice additions to manipulate what goes into the WEB-INF/classes folder, and generating the web.xml file. The WAR task is useful to specify a particular layout of the WAR file.

Since the WAR task is an extension of the JAR task, all attributes of the JAR task apply to the WAR task.

Attributes	Description
webxml	Path to the web.xml file
lib	A grouping to specify what goes into the WEB-INF\lib folder.
classes	A grouping to specify what goes into the WEB-INF\classes folder.
metainf	Specifies the instructions for generating the MANIFEST.MF file.

Continuing our **Hello World** Fax Application project, let us add a new target to produce the jar files. But before that let us consider the war task. Consider the following example:

```
<war destfile = "fax.war" webxml = "${web.dir}/web.xml">
  <fileset dir = "${web.dir}/WebContent">
    <include name = "**/*.*"/>
  </fileset>
  <lib dir = "thirdpartyjars">
    <exclude name = "portlet.jar"/>
  </lib>
  <classes dir = "${build.dir}/web"/>
</war>
```

As per the previous examples, the **web.dir** variable refers to the source web folder, i.e, the folder that contains the JSP, css, javascript files etc.

The **build.dir** variable refers to the output folder - This is where the classes for the WAR package can be found. Typically, the classes will be bundled into the WEB-INF/classes folder of the WAR file.

In this example, we are creating a war file called fax.war. The WEB.XML file is obtained from the web source folder. All files from the 'WebContent' folder under web are copied into the WAR file.The WEB-INF/lib folder is populated with the jar files from the thirdpartyjars folder. However, we are excluding the portlet.jar as this is already present in the application server's lib folder. Finally, we are copying all classes from the build directory's web folder and putting into the WEB-INF/classes folder.

Wrap the war task inside an Ant target *usuallypackage* and run it. This will create the WAR file in the specified location.

It is entirely possible to nest the classes, lib, metainf and webinf directors so that they live in scattered folders anywhere in the project structure. But best practices suggest that your Web project should have the Web Content structure that is similar to the structure of the WAR file. The Fax Application project has its structure outlined using this basic principle.

To execute the war task, wrap it inside a target, most commonly, the build or package target, and run them.

```
  <target name="build-war">
      <war destfile="fax.war" webxml="${web.dir}/web.xml">
          <fileset dir="${web.dir}/WebContent">
              <include name="**/*.*"/>
          </fileset>

          <lib dir="thirdpartyjars">
              <exclude name="portlet.jar"/>
          </lib>
```

```
        <classes dir="${build.dir}/web"/>
    </war>
  </target>
```

Running Ant on this file will create the **fax.war** file for us.

The following outcome is the result of running the Ant file:

```
C:\>ant build-war
Buildfile: C:\build.xml
BUILD SUCCESSFUL
Total time: 12.3 seconds
```

The fax.war file is now placed in the output folder. The contents of the war file will be:

```
fax.war:
    +---jsp //This folder contains the jsp files//
    +---css //This folder contains the stylesheet files//
    +---js //This folder contains the javascript files//
    +---images //This folder contains the image files//
    +---META-INF //This folder contains the Manifest.Mf//
    +---WEB-INF
        +---classes //This folder contains the compiled classes//
        +---lib //Third party libraries and the utility jar files//
        WEB.xml //Configuration file that defines the WAR package//
```