HowToBuildAndDeployCocoonWithMaven

Purpose

This how-to explains how to build Cocoon with Maven. An example for building a minimal webapp is provided for each option at the end of this document.

What is Maven?

Maven is a Java project management and project comprehension tool that is used in many projects around the world. The intent of Maven is to make intra-project development highly manageable in the hopes of providing more time for cross-project development. See Maven's Goals.

Prerequisites

- You must be familiar with the Cocoon's BuildSystem ;
- **Maven must be installed on your system.** For instructions on how to download and install Maven, follow this link: http://maven.apache.org/start/install.html

Integrate Maven with Cocoon

There is really only one step to integrate Maven with a project: the creation of a project descriptor. You may also create an ancillar *maven.xml* file which contains project specific goals, preGoals, and postGoals.

If you are using Maven for the first time or starting a new project you can use the GenApp plug-in to automate the creation of a Maven project tree.

maven -Dpackage=com.mycompany.app genapp

I'll present two approaches that were discussed on the user mailing list http://marc.theaimsgroup.com/?t=110114687600005&r=1&w=2.

Option 1 (the hard way...)

I used to build Cocoon this way before discovering Ralph's method. It consists mainly in manually tracking all the Cocoon's JARs and putting them into your Maven repository. Here the steps:

1. Build Cocoon with only the desired properties and blocks (as explained in INSTALL.txt) by typing *build* or *./build.sh*;
2. Move all the generated JARs (build/webapp/WEB-INF/lib) into your Maven repository;
3. Copy the generated Cocoon webapp folder (build/webapp) to your project source directory (e.g. projectA/src/webapp), except the JARs;
4. Add the JARs dependencies to your Maven project;
5. Build your project by typing *build war* or *./build.sh war*. It will download the dependencies into your local repository and create a WAR file.

The drawback of this solution is the need to upgrade all the Cocoon blocks and dependencies with each release (It could be a real pain, believe me!). Ralph's approach solves this problem.

Option 2 (recommended)

This method was introduced to me by RalphGoers (thanks Ralph!). It consists in building Cocoon and put the generated WAR file on your Maven repository.

1. Build Cocoon with only the desired properties and blocks (as explained in INSTALL.txt) by typing *build* or *./build.sh*;
2. Copy the generated WAR (build/cocoon-2.1.6/cocoon.war) into your Maven repository (you may want to rename it, for example cocoon-2.1.6.war);
3. Add the WAR dependency to your Maven project descriptor;
4. In maven.xml, add a preGoal element containing an unwar instruction for expending the specified WAR in your build directory;
5. Build your project by typing *build war* or *./build.sh war*. It will extract the Cocoon WAR into your build directory, add your stuff and create another WAR file.

Note that this means that the vast majority of the JARs you will be using will be brought in via the Cocoon WAR file, dramatically reducing the number of dependencies that have to be changed with each Cocoon update.

Use the examples

Option 1

cocoon-minimal-option1.zip

- Extract the zipped file in the location of your choice;
- Do steps 1 and 2. Steps 3 and 4 are done for you 🙂 ;
- Finally, do step 5.

Option 2

cocoon-minimal-option2.zip

- Extract the zipped file in the location of your choice;
- Do step 2 using the provided cocoon-2.1.6-minimal.war (or build your own);
- Finally, do step 5.

Note ⚠️

- The provided examples are known to work with Cocoon 2.1.6;
- If you use the Tomcat plug-in, start Tomcat then type *maven tomcat:deploy*.

Note ⚠️

- if you use the cocoon-war-2.1.6.war from ibibilio, you may want to use:

```
<dependencies>
    <dependency>
      <groupId>cocoon</groupId>
      <artifactId>cocoon-war</artifactId>
      <version>2.1.6</version>
      <type>war</type>
    </dependency>
  </dependencies>
```

you may want to use the maven.xml below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<project default="war"
    xmlns:jelly="jelly:core">

  <preGoal name="war:webapp">
    <jelly:forEach var="dependency" items="${pom.dependencies}">
      <jelly:if test="${dependency.getArtifactId().equals('cocoon-war')}">
        <jelly:if test="${dependency.getType().equals('war')}">
          <jelly:set var="cocoonWar"
              value="cocoon-war-${dependency.getVersion()}.war"/>
          <unwar src="${maven.repo.local}/cocoon/wars/${cocoonWar}"
              dest="${maven.build.dir}/${pom.artifactId}">
            <echo>Extracting ${cocoonWar}</echo>
          </unwar>
        </jelly:if>
      </jelly:if>
    </jelly:forEach>
  </preGoal>

</project>
```

Useful links

- The Cocoon 2.1 Build System: BuildSystem
- Maven: http://maven.apache.org/
- maven-proxy: http://maven-proxy.codehaus.org/
- maven-tomcat-plugin: http://www.codeczar.com/products/maven-tomcat-plugin/

EricJacob