Building your own Cocoon project using Ant

Table of contents

Table of contents

1. Building your own Cocoon project using Ant

Now that you've studied Cocoon and you are convinced it is worth spending your time on, you want to try your hand at your own project. There are several ways to start your own project. This is just one of them.

The following HOW-TO helps you to create your own project in a subdirectory of the Cocoon root. It also has its own subsitemap. The main advantage of this configuration is the fact that you can leave the original Cocoon root sitemap intact. Since it is set up in a pretty generic way and accommodating most types of generators, transformers and serializers, it provides a working environment that you don't have to understand thoroughly before you are able to use it. Instead, you can work the other way around. Once you understand a part of the root sitemap you can tune it to your needs.

Another advantage of this configuration is the strict separation of the files that are part of Cocoon and those that are part of your project. The separation allows you to update Cocoon without overwriting vital modifications. Or you can develop your webapp against different Cocoon distributions.
Note: This configuration will work with Cocoon 2.1.X distributions and maybe more recent. It will probably be more difficult to get this working with an older distribution.

1.1. Initial setup

The following steps explain how to set up this configuration. These steps will only be performed once. You will work from the commandline.

1. Get the Cocoon distribution. You can either download a release build or dive in and get the latest HEAD from SVN.
2. Make a separate project directory and put *build-cocoon-targets.xml* in this directory.
3. "Seed" the project by running the following on the commandline: ant -f build-cocoon-targets.xml -Dcocoon.distro.home=/path/to/your/cocoon/distribution
4. **Note:** Either you have ANT_HOME set to the appropriate directory or you use the full path to Ant.
   Note: The *cocoon.distro.home* is now set in *user.properties*.
   Note for Windows users: set the slashes as forward slashes.
5. Customize the *src/cocoon/local.build.properties* and remove the path variables in this file. Ant cannot substitute variables in property files, so ${...} variables would cause build errors. Remove the sections *build, build webapp, src, standalone demo, dir layout, tools, deprecated, ide, lib, dist, site, legal, gump*, so that there are no more ${...} and variables in this file.
6. Exclude all blocks you don't need in the top part of *local.build.properties*.
7. Exclude the samples and documentation too if you don't need them. The build process is time consuming.
8. Run the following command from the commandline: ant cocoon:get**Note:**This will only be done when changing the distribution or the settings in *local.build.properties*.
9. Run the following command from the commandline: ant webappThis builds the Java classes and copies the entire project to the tools tree.
   Note: classes are built with *debug* flag on.
10. Run the following command from the commandline: ant cocoon:runThis starts Jetty with Cocoon and your project in it.
    Note: Default Cocoon is configured to automatically reload files that are changed. This way Jetty can keep running while you develop your project. You only have to restart when the previous step has done any of the following:
    - changed any of the libraries,
    - compiled java classes.

11. Start up your favourite browser and enter the following: http://localhost:8888/You should now see the homepage of Cocoon.
12. Check your (now still empty) project in in your CVS.

This concludes the initial setup.

1.1.1. Example

If we execute the above steps on a Windows machine with the Cocoon distribution in /SVN/cocoon and your project in /projects/myprojects, the commands and the resulting information on your screen will look like this:

D:\projects>mkdir myproject D:\projects>copy build-cocoon-targets.xml myproject 1 file(s) copied. D:\projects>cd myproject D:\projects\myproject>\apache-ant-1.6.1\bin\ant -f build-cocoon-targets.xml -Dcocoon.distro.home=/SVN/cocoon Buildfile: build-cocoon-targets.xml seed-check: msg-seed-localprops: seed-dirs: [mkdir] Created dir: D:\projects\myproject\src\cocoon [mkdir] Created dir: D:\projects\myproject\src\cocoon\webapp [mkdir] Created dir: D:\projects\myproject\src\cocoon\xconf [mkdir] Created dir: D:\projects\myproject\tools\cocoon [mkdir] Created dir: D:\projects\myproject\tools\cocoon\webapp [mkdir] Created dir: D:\projects\myproject\src\java [mkdir] Created dir: D:\projects\myproject\lib seed-localprops: msg-seed-cvsignore: seed-cvsignore: msg-seed-userprops: seed-userprops: msg-seed-build: seed-build: [echo] Creating build.xml... seed: [echo] Done. [echo] [echo] The directory src/cocoon/webapp is created to hold your cocoon [echo] webapp resources. [echo] The directory src/cocoon/xconf is created to hold XConfPatch files [echo] to (optionally) modify the cocoon.xconf log.xconf web.xml and [echo] (root) sitemap.xmap [echo] [echo] From here: [echo] --------- [echo] You should now edit the file ./src/cocoon/local.build.properties to select [echo] only those optional components of Cocoon that your project needs. [echo] IMPORTANT: Remove the path-entries from that file! [echo] [echo] The build.xml can freely be extended for your project needs. [echo] [echo] To build a fresh Cocoon base for this project [echo] (when you updated the distro pointed to by -Dcocoon.distro.home) [echo] > ant cocoon:get [echo] [echo] To blend in your own project resources and classes: [echo] > ant webapp [echo] [echo] To test-run using the Jetty container: [echo] > ant cocoon:run [echo] BUILD SUCCESSFUL Total time: 1 second D:\projects\myproject>

1.2. Directory layout

Figure 1: Directory layout

In figure 1 you can see the resulting directory layout. We'll discuss what each of these entries contain:

- **lib** This directory contains project dependant jars. Not the Cocoon jars, since they are distribution dependent.
- **src/java** Your own Java classes and components go here.
- **src/cocoon/webapp** Add a directory <yourProjectName> here. This contains your Cocoon related files such as the sitemap, XSL and XML files.
- **src/cocoon/xconf** Your modifications to the Cocoon configuration files go here.
  Note: See xpatchusage for an explanation of how to create these modifications.
- **tools/cocoon/webapp** This will contain the resulting structure for Jetty to display. The Cocoon jars, copied from the Cocoon distribution are put here too in *WEB-INF/lib*.
- **build.xml** This will be your Ant buildfile. Modify it to your needs.
- **user.properties** and **src/cocoon/local.build.properties** contain some variables such as

cocoon.distro.home

1.3. During development

The *build.xml* contains several useful targets, which we will explain below. **Note** that this file includes *build-cocoon-targets.xml*. Some of the targets below are in that file, so don't delete it!

- **cocoon:get** Get Cocoon into your project. This target will compile the Cocoon distribution using your *local.build.properties* and put the result in *tools/cocoon/webapp*.
- **cocoon:unpatch** During *cocoon:get* the original versions of vital Cocoon configuration files such as *sitemap.xmap, cocoon.xconf, web.xml* and *logkit.xml* are copied to *tools/cocoon/unpatched*. This target allows you to copy the original versions to their respective locations. Very convenient when you changed an xconf file that cannot be applied.
- **webapp** Compile the Java classes in *src/java* and copy the relevant files in the source tree to the appropriate places under *tools/cocoon*. It also modifies the Cocoon configuration files using the xconf files.
- **cocoon:run** Start up Jetty with your webapp. Your project can be reached under http://localhost:8888/yourProjectName

The *build-cocoon-targets.xml* file can be found here.
Credits go to Marc Portier for the content and the Ant build script.

1.4. ToDo

- Use Cocoon 2.2 import mechanism
- figure out the difference between the Ant 1.5 setup and the Ant 1.6 setup. OTOH: is this still necessary?
- check English (grammar, style, US vs EN)
- check if all is (still) accurate -> it was as far as I can tell.
- Refer to this for adding it to Eclipse?
- create xpatchusage documentation

1. Comments

add your comments