Grouper UI Development Environment

This page is not updated. Please refer to this page

Grouper UI Development Environment

Introduction

This document is current as of release v1.2.0. For current info, please refer to this page

There are many ways to set up a development environment using a variety of open source and commercial tools and application servers. The environment described here is the one used to develop the Grouper UI - however, you are free to use whatever setup works best for you.

I am an active developer, so the directory layout and build scripts I use are designed to facilitate development as well as final deployment. We normally use Eclipse as a Java IDE, and so some choices I made are biased to *cope* with the way Eclipse works. - Gary Brown, UoB

Directory structure

In order to verify the extensibility of the UI, I have developed the Grouper UI and a custom (University of Bristol) version in parallel, using the same environment. A minimal implementation only requires a Grouper API installation in addition to a Grouper UI installation, however, any real world implementation will have site specific components as well:

Grouper UI Development at the University of Bristol, UK

Component	Description
grouper	The Grouper API
grouper-ui	The Grouper UI
uob-grouper-ui	Bristol customisations to the Grouper UI
i2mi-subject	Bristol implementation of the Subject interface. Sites may be able to use a generic source adapter provided with The Subject interface distribution e.g. an LDAP adapter

grouper and grouper-ui are separate modules in the I2MI CVS repository.

uob-grouper-ui and i2mi-subject are separate modules in the CVS repository at Bristol.

I have all the CVS checkout directories as subdirectories at the same level (to help Eclipse), though this is not an absolute requirement, i.e., :

```
GrouperComplete
   grouper
   grouper-ui*
   uob-grouper-ui*
   i2mi-subject
```

*Both directories contain a subdirectory *webapp* which itself has a directory structure that is consistent with a web application (see Architecture document)..

During development I may need to debug source code from any of the projects. I may also want to make code changes in the appropriate CVS checkout areas*. In my ideal development environment I would be able to edit any source files and instantly see changes in the web application. A typical build script for a web application might create the web application directory structure in a new *build* directory and then either copy or make into a WAR (web application archive file), and then deploy to a Servlet container e.g. Tomcat. Using this approach every change requires a build and potential restart of the web application. Admittedly Eclipse will allow an ant script to be called when source files are modified, however, this can be overkill for a simple change to a JSP.

*I could edit JSP and other files in the *build* or *deploy* directory, however, I would then need to copy the changes back to CVS - something I may well forget to do.

Setting Up Eclipse

In Eclipse I create one *project* and pull in the *java/src* and *lib* directories from each of the 4 projects listed above. I can then set a single output directory where compiled Java classes are placed whenever I save a Java source file. JSP and other *content* files are trickier since they are saved *in situ* and not compiled to a separate destination. Normally I will be working on *either* the core Grouper UI *or* on Bristol customisations . In the Grouper UI *build.properties* file I can elect to have the *webapp* directory of grouper-ui *or* uob-grouper-ui be the web application root (configured in Tomcat)*. I manually configure Eclipse to compile Java classes to the appropriate *webapp/WEB-INF/classes* directory.

*Actually, any directory can be configured to be the web application root - I always choose either of the ones indicated when developing.

Any changes I make to the *local* JSPs are immediately picked up by Tomcat, however, I would need to run an ant script to obtain changes from the other project i.e. *uob-grouper-ui* to *grouper-ui.* Most sites which are not involved in the development of the Grouper UI should set *<institution>-grouper-ui/webapp* to be their web application root. If working with *Tomcat* and the build.properties *deploy* properties are set, the build script will automatically install your webapp on Tomcat such that Tomcat reads files from your *work area*.

A disadvantage of this approach is that it *pollutes* the CVS checkout area for one module with those from another, and I may be tempted to edit a file in the wrong location (though hopefully they are in different subdirectories). Assuming that site-specific changes are always in distinct subdirectories then on Unix it may well be possible to set up symbolic links from grouper-ui to, say, uob-grouper-ui.

Some changes e.g. adding new JAR files, modifying resources, changing Struts / tiles configuration files will always require a build and a web application restart.

The Ant Script

The following targets are available:

```
>ant help

 Buildfile: build.xml        help:

[echo] Please ensure you have read the documentation -
[echo] and created a build.properties file based on the template provided
[echo] The following targets are available - type the appropriate name:
[echo] 1) default
[echo]      Simply builds, without cleaning, to the default.webapp.folder
[echo] 2) nice
[echo]      Attempts to stop the Tomcat webapp before building.
[echo]      Attempts to start the webappp afterwards
[echo] 3) clean
[echo]      Always removes the webapp.class.folder. May remove the
[echo]      webapp.folder if webapp.folder.cleanable=true
[echo]      On Windows this may fail as Windows tends to lock files
[echo] 4) niceclean
[echo]      Combination of nice and clean
[echo] 5) dist
[echo]      Cleans and then builds to subfolder of dist.home
[echo] 6) war
[echo]      Does dist and then makes a WAR file
[echo] 7) resources
[echo]      Does not compile Java classes but 'refreshes' resources in
[echo]      webapp.class.folder
[echo] 8) niceres
[echo]      Does not compile Java classes but 'refreshes' resources in
[echo]      webapp.class.folder and restarts webapp
[echo] 9) help
[echo]      Displays this menu
[echo] 10) endhelp
[echo]      Subsequent invocation of ant with no target will run
[echo]      'default' rather than help
[echo] 11) starthelp
[echo]      Subsequent invocation of ant with no target will run 'help'
[echo] 12) html
[echo]      Generate Javadoc - you must have done a 'default' build previously
[echo] 13) exit
[echo]      Exit this menu without executing another target
[input] Make your choice (default)>
```

The *nice* targets will only work if you are using Tomcat and have configured the deploy properties in build.properties, and have installed catalina-ant.jar with Ant.

See Customising the Grouper UI: Customising the Build Process for details on how to customise the build process.