Synchronizing templating of Author and Public instances



Your Rating:  Results:  124 rates

For those who are familiar with ANT here is a simple ANT-buildscript to sync templating of publish and author instance.
Everything is straight forward and you should easily be able to adapt it to your structure.

1. install ant (http://ant.apache.org/) or run within Eclipse
2. copy the snipped to sync.xml
3. adapt "sourceDir" and "targetDir" to match you installation
4. make backup of author and publish instance
5. run ant -f sync.xml
6. restart publish instance (if you changed xml-files)

I use this script within my development environment (Eclipse) and a similar script to sync my development author instance with my production instances (publish and author) on a linux server (connection with ssh).

```
<!--===========sync.xml=================--> <?xml version="1.0" encoding="UTF-8"?>  <project name="syncPublic"
default="syncLocal" basedir="."> <property name="sourceDir" value="C:/Programme/magnolia/author/webapps
/magnolia"/> <property name="targetDir" value="C:/Programme/magnolia/public/webapps/magnolia"/> <property name="
docroot" value="docroot"/> <property name="templates" value="templates"/> <property name="dialogs" value="
repository/config/dialogs"/> <property name="paragraphInfo" value="repository/config/paragraphInfo"/> <property
name="templateInfo" value="repository/config/templateInfo"/>  <target name="syncLocal" depends="syncDocroot,
syncTemplates,syncDialogs,syncParagraphInfo,syncTemplateInfo"> </target>  <target name="syncDocroot"> <echo
message="${targetDir}/$\{docroot}"/>
<sync todir="$\{targetDir}/$\{docroot}">
<fileset dir="$\{sourceDir}/$\{docroot}" casesensitive="yes">
</fileset>
</sync>
</target>

<target name="syncTemplates">
<sync todir="$\{targetDir}/$\{templates}">
<fileset dir="$\{sourceDir}/$\{templates}" casesensitive="yes">
</fileset>
</sync>
</target>

<target name="syncDialogs">
<sync todir="$\{targetDir}/$\{dialogs}">
<fileset dir="$\{sourceDir}/$\{dialogs}" casesensitive="yes">
</fileset>
</sync>
</target>

<target name="syncParagraphInfo">
<sync todir="$\{targetDir}/$\{paragraphInfo}">
<fileset dir="$\{sourceDir}/$\{paragraphInfo}" casesensitive="yes">
</fileset>
</sync>
</target>

<target name="syncTemplateInfo">
<sync todir="$\{targetDir}/$\{templateInfo}">
<fileset dir="$\{sourceDir}/$\{templateInfo}" casesensitive="yes">
</fileset>
</sync>
</target>

</project>
<!--========================-->
```

Synchronize Development environment with production server

Please make backup before trying and verify the settings

run
`ant -f to_Server.xml`

Another approach - ant script for copying of templates

There are two ways to work with modules like this (examples):

- As a developer - working with the magnolia source tree in `c:\projects\magnolia-all`
  - `c:\projects\magnolia-all\magnolia` <-- original webapp module
  - `c:\projects\magnolia-all\magnolia-module-mine` <-- our module
- As a user - working with the magnolia binaries in
  - `c:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\magnoliaAuthor`
  - `c:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\magnoliaPublic`
  - `c:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\magnolia-module-mine`

When starting your own module, copy the directory structure from magnolia-all\magnolia (this is the webapp module of magnolia).

The attached build.xml is put in magnolia-module-mine

For example, your templates or JSPs should be kept in `c:\projects\magnolia\magnolia-module-mine\src\main\webapp\templates\jsp`

Now you must change the variable "magnolia-dir" inside build.xml to the one you want to copy resources into. This can be either:

- `<property name="magnolia-dir" value="magnoliaAuthor" />` for copying into the authoring environment
- `<property name="magnolia-dir" value="magnoliaPublic" />` for copying into the public environment
- `<property name="magnolia-dir" value="magnolia/src/main/webapp"/>` for copying into a development environment

Installing the module

1. Checkout/copy your magnolia-module-mine next to your magnolia instance
2. Set the magnolia-dir property in build.xml to be the magnolia instance where you want the module
3. Run ant (default goal) inside magnolia-module-mine
4. Repeat for each instance

If your module contains any classes, you need to manually copy these into magnolia's classpath.

Pretty simple and straightforward if you know ant. I'm sure it could be done in a more elegant way, and even be incorporated into maven2, but I haven't been bothered with finding out how this is done.

There is also some functionality if you have your own pom.xml inside `magnolia-module-mine\poms` which will overwrite the poms in magnolia webapp and magnolia-project, but this is for advanced users only 😎

> *In its previous incarnation on JspWiki, this page was last edited on Feb 9, 2007 11:00:23 AM*