Customer Project Setup



Your Rating:  Results:  23 rates

- Best practice project setup with a separate Magnolia module
  - Structure
  - Setting up access to the enterprise maven repository
  - Materialize the webapp
  - Assign enough memory
  - Configuration
  - Copying the jsp from the module to the webapp on changes
  - Building a clean deployable war
  - IDE integration

Best practice project setup with a separate Magnolia module

Here you find a demo project setup myproject.zip.

The ZIP file uses Magnolia version 4.0.1. When changing that to a more recent version, another repository for the Enterprise components is needed: in `pom.xml` change http://repo.magnolia-cms.com/restricted into http://repo.magnolia-cms.com/enterprise.

Structure

- parent pom
  - mywebapp (webapp inheriting from the ee webapp)
  - mymodule (copy of our samples module)

Setting up access to the enterprise maven repository

See Maven Setup.

Materialize the webapp

The webapp has no content (no web.xml, ..). This is based on the fact the webapp has a dependency to the Magnolia webapp which contains all relevant files. So unless you need any

1. call `mvn install` in the root of the project (to install all poms)

2. call `mvn war:inplace` in the webapp folder (to overlay the empty webapp with the Magnolia webapp)
3. in case you use the maven support of your IDE delete WEB-INF/lib to ensure that you don't get conflicts

Assign enough memory

When not using `magnolia_control.sh` to start the application server make sure when starting app server (inside or outside of your IDE) you assign it enough of memory, the `-Xmx512M` and `-XX:MaxPermSize=256m` are the values recommended for production and should be more then enough for development

Configuration

Please visit our configuration page explaining how the basic configuration works. In `WEB-INF/config/default/magnolia.properties` you might like to change the values of:

- `magnolia.repositories.home` to a value outside the webapp to not loose the data on redeploys
- `magnolia.bootstrap.samples` to false if you want a clean repository without samples
- `magnolia.update.auto` to true if you don't want to click you through the installation screens when ever you flush the repository

Copying the jsp from the module to the webapp on changes

Normally the jsp in mgnl-files get extracted in the installation process of a module. But while developing they remain in the classpath (jar) because the installation is not triggered. A convenient solution is to use an ant script which copies the files to the webapp upon changes. In Eclipse you trigger this by adding an ant builder to the project.

An example build.xml file can be found in the mymodule folder.

Building a clean deployable war

⚠️ Don't build the war file in the webapp you use in the IDE. You have probably run `mvn war:inplace` or you have an already bootstrapped repository folder.

- make a fresh checkout
- call `mvn install`

IDE integration

- Eclipse