DotCMS v2.2 upgrade

MLH Code Updated : 12/16/2012
DotCMS Code : Tag v2.2

The upgrade for DotCMS was quite time consuming. First, because there were so many API changes this time around. Second, because the MLH and DotCMS integration has been quite extensive. There are several ways to go about upgrading DotCMS and typically you could just update the codebase and move forward. Not so with the MLH integration, since there are many Java and Velocity files that have been overridden. So what you end up having to do is to update the core code while leaving the MLH additions. The best way to do this is to keep the previous dotcms core code in another project, preferably in the same workspace. This way you can view the core changes from one version to the next and only update that code while leaving the MLH update alone.

It doesn't look like the DotCMS team widely uses MSSQL for development or testing. So there were a few problems with the database upgrade. I have fixed the few problems areas and the upgrade will be seemless. After the initial upgrade you will no longer need to worry about upkeep of the old database upgrade files - just the new ones :).

LOGGING

You can now just adjust the logging level from the com.dotcms.config plugin - there is already a log4j.xml file located there.

JAR EDIT

I have removed the 0.19.x version jar from DotCMS altogether - WEB-INF/lib and the ES plugin. I have replaced it in WEB-INF/lib with the 0.20.1 version - which has our ES System Properties fix.

NOTE: This means that the indices's the ES Plugin currently uses must be upgraded to 0.20.1 to ensure proper indexing. I tried to use the 0.19.9 version as the remote index but kept getting errors - just use 0.20.1 and above.

GitHub

We are using a tag version, that is what is currently set up as the working copy in the git project. Using Git at this point forward makes the most sense. If need be I can commit all code to my fork and then transfer ownership of the project to another github organization - as long as I have admin rights - or another GitHub user. Currently GitHub public forks cannot be made private. So you will have to either contact GitHub support or use this approach https://help.github.com/articles/duplicating-a-repo . After that you can delete the public fork. However it works best is okay for me.

Compiler Errors

You should only see these errors in your project :

The type FolderWebAPI is already defined FolderWebAPI.java
The type FileTool is already defined FileTool.java
The type NavigationWebAPI is already defined NavigationWebAPI.java
The type JSONTool is already defined JSONTool.java
The type ContentletAjax is already defined ContentletAjax.java

The reason should be obvious, but just so everyone knows - these are the Java files that MLH has overriden and both are on the classpath. Since the plugin projects are first in order they take priority. So in effect, the src/ of every Plugin that defines it will have to come before the DotCMS src/ and src-conf/. The project will have the eclipse .project committed, so this should solve any configuration problems; save for the JDK version - you might have to change that.

Configuration

All configuration is now done in the com.dotcms.config plugin. No need to edit any core configuration files anymore, just add the file to the plugin in the file structure needed. One current caveat with this is the fact that you need to have the whole file for editing as they get replaced not updated, at least for the files needed for Tomcat. All internal configuration files will comment out the updated portion and add your settings.

For development I have just been using the default /assets location, copied the 'dotcms_assets' to the dotCMS/dotCMS/ directory and renamed it to just 'assets'. If the assests folder were in the repository I would say that we would just update the com.dotcms.config plugin to the appropriate path, using this property:

ASSET_REAL_PATH

This will allow you to have the assets in an external folder. However, since we have to copy over the 'dotcms_assets' manually anyway, might as well put it where DotCMS expects it by default - keep everything nice and tidy. In production, of course we would use the property to set the path to an external directory.

NOTE: I would recommend, at this time, that for all database, apache/webroot, and dotcms_assets backups to be taken from QA instead of dev(unless otherwise directed). You should still get the webroot/assets and dotcms source from the repository. Getting SVN access and checking out the needed projects is out of the scope of this document.

Apache Proxy

The MLH DotCMS expects a Proxy to handle delivery of some of the static content. For this reason it is required that you have Apache installed and configured properly. It would be best to gather the Apache configuration files from either QA or DEV - however it seems QA is a more stable location for getting Apache configuration files. Installing and configuring Apache is out of the scope of this document. However, you do need to have the apache-webserver project checked out from the repository and the assets/ folder underneath put into Apache/webroot. You must get the other needed files from QA Apache/webroot or on DEV.

Build

The way we have to build this is a bit off, because of the need to override core java files. I have been able to break it down and make it a lot easier. I have included a few new, well copies, of the original build.xml. They are called – build-custom, clean, and deploy. To build the project you follow this work flow :

build-custom.xml (does a full clean then a build) - right click --> Run As --> 1 Ant Build
plugins/mlh.dotcms.admin.scripts/build.xml - right click --> Run As --> 1 Ant Build
deploy.xml - right click --> Run As --> 1 Ant Build

NOTE : You will need to run the deploy.xml twice - as there are some parsing problems with the DRW.xml. Running a second time will resolve this problem, or else you end up with an error when trying to load a page. In the future it would be nice to have a mlh-build.xml that goes out and does everything provided above in a single call, including calling deploy twice.

I also added a clean.xml if you wanted to just clean the project. Just use this procedure : right click --> Run As --> 1 Ant Build.

Performing the Upgrade

The first thing you need to do is take a backup of the dotcms database, just in case. Then you will need to run the SQL Database Prepare script, this will ensure the database is ready for the upgrade. After the script is done running you can then run the application so we can finish up the upgrade.

You will need to ensure this file is configured properly for your database – it also includes the MSSQL Driver block.

dotCMS/plugins/com.dotcms.config/ROOT/tomcat/conf/Catalina/localhost/ROOT.xml

At this point the project should be fully configured and ready to go - or ensure all properties files in the Plugins are correct. You will build the application using the build steps – which doesn't affect the way you deploy it (according to the current set up – from my understanding). You will then run the application as you normally would. The application will go into upgrade mode and make sure all database objects are fully updated. There will be a few errors at the end, but they are just Warnings so you can just ignore them - something about "escalation" not being in a table. The reason for this error is because they use the same bit of code to update that table and at first it expects the "esca.." column to be there but that database statement comes later - for some reason or another.

NOTE: Please be patient with the database upgrade process - it can take some time to perform depending on your hardware.

After DotCMS is booted up go ahead and shut her down. We will need to run the SQL Database Cleanup script that will make sure the database is cleaned up from the upgrade and previous versions. Once the database script is finished go ahead and restart DotCMS. Now we will need to login to DotCMS as an Administrator. Go to CMS Admin --> CMS Maintenance and click on the Index tab. From here click on "Rebuild Index" or “Re-Index”. This will rebuild your index and will take some time to perform. After the indexing is complete you should be able to use your Upgraded DotCMS Instance!

Run in Eclipse

I have created a dotcms.launch file that you should be able to use to run dotcms inside Eclipse, for general development or debugging. You should be able to load the .launch file from the Run --> Run Configurations dialog box from within Eclipse. From there you can edit the .launch configuration, if you need to change the JVM arguments or other settings. You will need to change the locations of the dotCMS project inside Arguments tab in the VM Arguments text box.

Suggested JVM Arguments (already part of the dotcms.launch file) :
-server
-Xms768M
-Xmx2304M (or as high as you can go)
-XX:PermSize=256m (this depends on use - could be as high as 512 but shouldn't need to be higher)
-XX:+UseConcMarkSweepGC

Known Velocity Errors

There is a text file that has the two errors, in the same directory this file is, that I have seen when running over the DotCMS pages. They don't cause any display issues and no errors in the browser console. I know one issue to be a NULL value when it expects to have a value. Basically, we get a NULL exception because the content has an empty value - but the content is supposed to(so no real issue there). The other issue is due to a #parseContainer error, but I cannot find any locations where we are parsing anything that would blow up. I figured there should be some documentation on this, there are also a few 404 errors lingering here and there. I was using QA for the update data, so maybe that was the cause.

Changed DotCMS Source Files

I had to edit two files so that we could have the MLH environment working properly. Here is the information on that:

/dotCMS/src/com/dotcms/content/elasticsearch/business/ESContentletIndexAPI.java

This file was updated to have the correct import statement, ES changed where the class was located. We now import the correct class, you will get a compiler error otherwise:

import org.elasticsearch.action.admin.indices.exists.indices.IndicesExistsRequest;

/dotCMS/src/com/dotmarketing/startup/runonce/Task00785DataModelChanges.java

This file was updated so that we had the correct SQL Statement for MSSQL. SQL Server was not liking the insert at line 96. I have changed it so that it handle that without a problem. Should be :

```
insert into inodeskill(inode) (select inode from inode where type in('htmlpage','links','contentlet',
        'containers','template','file_asset') and (identifier is null OR (identifier not in(select inode from
        identifier))))
```