Building DSpace From Source

Contents

See Also

The Build Cookbook if you are adding your own code to a binary distribution, or want to make use of module overlays.

Prerequisities

You will need to have Subversion and Maven installed (in addition to Java and the other DSpace prerequisites.)

Check out DSpace from SVN

To check out DSpace, first determine the version of DSpace you want to install.

A few tips on how the DSpace SVN repository is organized / managed:

- */tags/* - Tagged versions represent*stable, official releases* of DSpace software. If you want to install a supported version of DSpace, find the appropriate tag to checkout.
- */branches/* - Branches are used for development of new *minor, bug-fix* releases (e.g. 1.4.2, 1.5.1). They should be considered semi-stable, but are not recommended unless you plan to help in development work.
- */trunk/* - Trunk is used for development of the next *major* version (e.g. 1.4, 1.5, 2.0). It should be considered *highly unstable*, and is not recommended unless you plan to help in development work.

Once you've chosen your version, check it out as follows (the below example is for DSpace 1.5):

```
svn co https://scm.dspace.org/svn/repo/dspace/tags/dspace-1_5 /some/where
```

Where "/some/where" is a path on the local file system that may be relative or absolute.
This will check out all of the DSpace modules.

Initial Build and Installation

To build DSpace (with PostgreSQL as the database):

Navigate into the directory called "dspace" within your svn checkout.

```
cd /some/where/dspace/
```

Execute Maven using the following command

```
mvn package
```

Note, one previously had to execute Maven with "assembly:assembly" as an option, this is no longer necessary. Other options for executing maven include designating the database to be used, this can be executed wih the following additional option.

```
mvn -Ddb.name=[postgres|oracle] package
```

This might take a while as required JARs etc are downloaded.

Now you should have a DSpace build in

```
dspace/target/dspace-1.5-SNAPSHOT.dir
```

. Set up a PostgreSQL database as usual. Edit `dspace/target/dspace-1.5-SNAPSHOT.dir/config/dspace.cfg` as required. Run:

```
ant fresh_install
```

You may see errors like this, but these don't matter:

```
[java] log4j:ERROR Could not read configuration file [file:config/log4j-console.properties].
[java] java.io.FileNotFoundException: file:config/log4j-console.properties (No such file or directory)
[java]     at java.io.FileInputStream.open(Native Method)
[java]     at java.io.FileInputStream.<init>(FileInputStream.java:106)
....
```

as long as you see BUILD SUCCESSFUL you should be OK.

In your DSpace install directory (by default `/dspace`), you should see a `webapps` dir. Copy these to your Tomcat's `webapps` dir.

Now http://localhost:8080/jspui/ should be the 'classic' DSpace UI. http://localhost:8080/xmlui/ should be the XML UI.

Development

Editing the source is now possible using any tools you like.

Using Eclipse is actually very easy if you're on 3.3 and have Subclipse installed. Just create a Java project, specify 'create project from existing source' and point it at `/path/to/src`, i.e. the directory with `dspace`, `dspace-api` etc. in it. Eclipse picks up the source directories and JARs, and even the SVN source control info. You might like to change Eclipse's build directory and remove duplicates and `dspace-*.jar` from the JARs in the build path, but as far as source editing with code completion goes you're good.

To deploy your changes (you'll probably want to set up a script to do this!)

```
$CATALINA_HOME/bin/shutdown.sh
cd /path/to/your/src/dspace
mvn package
cd target/dspace-1.5-SNAPSHOT.dir
ant -Dconfig=/dspace/config/dspace.cfg update
rm -r $CATALINA_HOME/webapps/dspace-*
cp -r /dspace/webapps/* $CATALINA_HOME/webapps/
$CATALINA_HOME/bin/startup.sh
```

Warning: the `mvn package` will overwrite the dspace.cfg in `dspace-1.5-SNAPSHOT.dir/config`.

Keeping up to date with DSpace SVN

In the top level of your checked out DSpace (i.e. the `/path/to/your/src` dir) do:

```
  svn up
```

You should see messages about what's being updated. If you see 'C' or messages about conflicts by any of the files, you will need to go to the files where there were conflicts and resolve them. See the section entitled "Resolve Conflicts (Merging Others' Changes)" in the SubVersion documentation.

Creating a patch file

NB: Read Code Contribution Guidelines before submitting a patch

To create a patch for all changes, run

```
svn diff -u > mypatch.txt
```

To create a patch for only certain files, run

```
svn diff -u path/to/files > mypatch.txt
```

Applying a patch file

The easiest way to apply a patch is by using the Linux `patch` command. For information on what the `patch` command does and all of its options, take a look at `man patch` from a Linux machine.

TODO: More detail needed

In general, you most likely will want to run a command *similar* to the following from your [dspace-source] directory (but see the *Important Hints* below, before running anything!):

```
  patch -p0 < mypatch.txt
```

Important Hints:

- It's *highly recommended* to add the `--dry-run` option initially, in order to test your command first, since not all patches are created equal!
- The `--verbose` option is sometimes helpful in understanding what applying the patch is actually doing.