DSpace Release 1.4.0 Documentation

Note: Some details below relate to DSpace version 1.1.1.* [More up-to-date documentation|http://www.dspace.org/current-release/latest-release/] is available though there's still some useful information e.g. about configuring Tomcat for SSL below.

Installation

Acquiring DSpace, prerequisites

Pre-install

Get DSpace from dspace.sourceforge.net.

1. A unix-like operating system (Linux, Solaris, BSD, HP/UX, ...)
2. A java developer environment (SDK) - we used Sun's 1.4.2 SDK
3. ANT, the Java compile tool
4. A servlet container/http server that can do SSL (Secure Socket Layers) in this course, we opted for standalone Tomcat. Version 5.0.x seems to work fine on Linux, but not always on Solaris. There you might be better off with the latest 4.1.x release. Other alternatives include Jetty, JBoss and similar.
5. The Postgres open source database server, version 7.3 or higher Currently DSpace can't work with any other database yet, although this is likely to change in the future.
6. Three separate java library packages:
   - Javabeans Activation Framework
   - Java Servlet 2.3 and JSP 1.2
   - Javamail API

Ideally, you want to download these files to a separate download directory so you can refer to these files later.

We assume some knowledge of the unix operating system so you're supposed to know how to install software in a permanent location, or how to unpack software to a temporary directory. If you don't feel comfortable with this, please get the help from a Unix sysadmin.

Make sure you have a "dspace" user on your system.

The following notes contain examples that were specific to the Mandrake Linux operating system on the course PCs. The actual location of files might be different on your system. The "export" syntax for setting variables is bash shell specific. If you use a different shell, it's up to you to find out how to set and export environment variables.

Postgres

On the course systems, Postgres was pre-installed from binary. Not all users will be so lucky - there are no binary packages of Postgres 7.3+ for Solaris, for example.

Once installed, you need to make sure you can connect to the database server from the local machine. To do this you may have to edit `pg_hba.conf` (wherever this is installed on your system).

You will probably also have to edit the postgresql.conf file to allow tcp/ip connections to the database server.

For example, on our Mandrake system, this meant (as root)

`vi /var/lib/pgsql/data/postgresql.conf`
add the line:
`tcpip_socket = true`
restart postgres with `/etc/init.d/postgresql restart`

On a Solaris9 system where PostgreSQL was installed with the default settings, these files is likely to be in /usr/local/pgsql/data (the directory that was created during "initdb" - see the PostgreSQL admin guide for more information).

On Debian Linux, the Postgres configuration can be found in `/etc/postgres`.

Creating the DSpace user and database

1. Become the right postgres user for your system, usually called "postgres":

   ```
   su - postgres
   ```

   (unless you execute this su from user root, this will ask for a password)
2. ```
   createuser -d -A -P dspace
   ```

   Make sure to remember the password you assign to the dspace user, you'll need this in the next step, and in the dspace config later.
3. ```
   createdb -U dspace -E UNICODE dspace
   ```

If there are problems, such as a message such as

createdb: could not connect to database template1: FATAL: IDENT authentication failed for user "dspace"

You'll need to do some work on your Postgres security settings. See Postgresql Configuration. You may also be interested in doing some Postgres Performance Tuning, since the default install has a conservative configuration.

Java SDK

The Java SDK can be installed as binary either in plain binary format or as a RPM for some linux systems. In any case, the Java SDK is all contained within one single directory. In the course example, this directory was /usr/java/j2sdk...

Set

```
JAVA_HOME
```

with:

```
export JAVA_HOME=/usr/java/j2sdk...
```

making sure that you use the directory the software is contained in on your system.

On a typical Solaris9 system, Java will already be installed. In this case,

```
export JAVA_HOME=/usr/j2se
```

Ant installation from source

Your system might come with Ant packages preinstalled, in which case you can skip this step. If you need to install Ant from source, you can follow these steps:

1. Extract the ant archive somewhere
2. Change to the ant directory you just created
3. Set ANT_HOME to this directory with:

```
export ANT_HOME=$(pwd)
```

4. Add ant to your PATH: export PATH=$PATH:$ANT_HOME/bin

Tomcat

Assuming you downloaded Tomcat as binaries from http://jakarta.apache.org, do the following to install it:

1. Extract the tomcat archive in a convenient directory, for example
   `/usr/local`
   or
   `/opt`
2. Change to this directory
3. `export CATALINA_HOME=$(pwd)`
4. You may have to run the following command:
   `export JSSE_HOME=$JAVA_HOME/jre/lib`
   (this gives Tomcat access to the java SSL libraries)

DSpace

1. Unzip/untar the dspace source package in a temporary directory with tar zxvf dspace...tar.gz
2. cd to the dspace source directory you just created
3. Extract the javamail, javabeans activation framework and servlet jars to a temporary directory.
4. Copy all the .jar files from these into `lib`
5. Edit dspace configuation file `config/dspace.cfg`:
   `dspace.url = https://localhost:8443`

   (8443 is the port the Tomcat SSL uses by default. This can be changed in the Tomcat config if you want)
   a. `dspace.hostname = localhost`

      For a real dspace system, you want to change these to use the publicly visible hostname of the server running DSpace.
   b. Add the correct database user/password information.

6. Copy the postgres jdbc driver into `lib`. On Solaris, with a stock Postgres 7.3 install, this would be, for example:

   ```
   cp /usr/local/pgsql/share/java/postgresql.jar lib/
   ```

   If you built postgresql from source (with the --with-java option) Then postgresql.jar would have been built at that time.

7. In the dspace source dir, do

   ```
   ant; ant fresh_install
   ```

Configuring Tomcat for operation

1. Connect dspace webapps to Tomcat. Go to

```
$CATALINA_HOME/webapps
```

2. Make symbolic links into the dspace install directory:

```
ln -s /dspace/jsp dspace
ln -s /dspace/oai dspace-oai
```

3. Make Dspace the main Web UI webapp: (still in the webapps directory)

```
mv ROOT ROOT.bak
ln -s /dspace/jsp ROOT
```

4. SSL setup. Generate a key for SSL (as root):

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```

and use the password:

```
changeit
```

Edit `$CATALINA_HOME/conf/server.xml:` find the example ssl server connector config and remove the

comment tags surrounding it ()

Add comment tags around the "normal" (non-SSL) Tomcat connector.

5. If you're using Tomcat 4.1.14 or higher, you may have to configure it to allow symbolic linking. For this, Add the following in server.xml inside the main "`<Host>`" section (so before the main "`</Host>`" tag):

```
<Context path="" docBase="ROOT" debug="0">
     <Resources className="org.apache.naming.resources.FileDirContext" allowLinking="true" docBase="" />
</Context>
```

6. Start Tomcat:

```
$CATALINA_HOME/bin/startup.sh
```

If you now go to https://your.dspace.server:8443/ you should see the DSpace you just installed. Replacing "your.dspace.server" with the hostname of your dspace server. This would be "localhost" in our example.

Creating an administrator account

To be able to go into the admin section (following earlier conventions, that would be https://your.dspace.server:8443/admin) you'll need to create an admin user from the shell. To do this, go to the dspace directory, and execute:

```
./bin/create-administrator
```

This will ask you for an email address, name and password, and set up the administrator.

Configuration and Customization

There are a number of ways to customize your DSpace installation. The most straightforward
is to edit the files in

```
/dspace/config
```

(if you installed dspace in

```
/dspace
```

, which is the default).

You can also edit the .jsp files (Java Server Page), and replace both the static HTML and the code in them. You can go even further and edit the sourcecode itself, or add more code to the system. However. All these methods are likely to produce a DSpace system which is very different from the main DSpace project, and is likely to break when you upgrade to the next DSpace version.

The dspace.cfg file

The primary way of configuring DSpace is to edit the `dspace.cfg`. You'll definitely have to do this before you can operate DSpace properly. dspace.cfg contains basic information about a DSpace installation, including system path information, network host information, and other things like site name.

The default `dspace.cfg` is a good source of information, and contains comments for all properties. It's a basic Java properties file, where lines are either comments, starting with a '#', blank lines, or property/value pairs of the form:

```
property.name = property value
```

For more information, see the ConfigurationKeyIndex page.

Wording of E-mail Messages

Sometimes DSpace automatically sends e-mail messages to users, for example to inform them of a new workflow task, or as a subscription e-mail alert. The wording of emails can be changed by editing the relevant file in /dspace/config/emails. Each file is commented. Be careful to keep the right number 'placeholders').

The Dublin Core and Bitstream Format Registries

The /dspace/config/registries directory contains two XML files. These are used to load the initial contents of the Dublin Core type registry and Bitstream Format registry. After the initial loading (performed by ant fresh_install above), the registries reside in the database; the XML files are not updated.

Currently, the system requires that every item have a Dublin Core record. The exact Dublin Core elements and qualifiers that are used can be configured by editing the Dublin Core registry. This can either be done at install-time, by editing /dspace/config/registries/dublin-core-types.xml, or at run-time using the administration Web UI. However, note that some elements and qualifiers must be present for DSpace to function correctly since they are used for various purposes by the code. Details are in the relevant .xml file.

The bitstream formats recognized by the system and levels of support are similarly stored in the bitstream format registry. This can also be edited at install-time via /dspace/config/registries/bitstream-formats.xml or by the administation Web UI. The contents of the bitstream format registry are entirely up to you, though the system requires that the following two formats are present:

- Unknown
- License

Configuration Files for Other Applications

To ease the hassle of keeping configuration files for other applications involved in running a DSpace site, for example Apache, in sync, the DSpace system can automatically update them for you when the main DSpace configuration is changed. This feature of the DSpace system is entirely optional, but we found it useful.

The way this is done is by placing the configuration files for those applications in /dspace/config/templates, and inserting special values in the configuration file that will be filled out with appropriate DSpace configuration properties. Then, tell DSpace where to put filled-out, 'live' version of the configuration by adding an appropriate property to dspace.cfg, and run /dspace/bin/install-configs.

Customizing the Web User Interface

The Web UI is implemented using Java Servlets which handle the business logic, and JavaServer Pages (JSPs) which produce the HTML pages sent to an end-user. Since the JSPs are much closer to HTML than Java code, altering the look and feel of DSpace is relatively easy.

To make it even easier, DSpace allows you to 'override' the JSPs included in the source distribution with modified versions, that are stored in a separate place, so when it comes to updating your site with a new DSpace release, your modified versions will not be overwritten.

The JSPs are stored in /dspace/jsp. Place your edited version of a JSP in the /dspace/jsp/local directory, with the same path as the original. If they exist, these will be used in preference to the distributed versions in /dspace/jsp. For example:
DSpace default Locally-modified version
/dspace/jsp/community-list.jsp /dspace/jsp/local/community-list.jsp
/dspace/jsp/mydspace/main.jsp /dspace/jsp/local/mydspace/main.jsp

Heavy use is made of a style sheet, in /dspace/jsp/styles.css.jsp. If you make edits, call the local version /dspace/jsp/local/styles.css.jsp, and it will be used automatically in preference to the default, as described above. Be sure to remove the localVersion code to avoid a loop!.

Fonts and colors can be easily changed using the stylesheet. The stylesheet is a JSP so that the user's browser version can be detected and the stylesheet tweaked accordingly.

The 'layout' of each page, that is, the top and bottom banners and the navigation bar, are determined by the JSPs /dspace/jsp/layout/header-**.jsp and /dspace/jsp/layout/footer-**.jsp. You can provide modified versions of these (in /dspace/jsp/local/layout, or define more styles and apply them to pages by using the "style" attribute of the dspace:layout tag.

Custom Authentication Code

Since many institutions and organizations have exisiting authentication systems, DSpace has been designed to allow these to be easily integrated. To do this, you can provide a custom class implementing the Java interface org.dspace.app.webui.SiteAuthenticator. These methods are invoked when various authentication-related events occur in the Web user interface.

The basic authentication procedure in the DSpace Web UI is this:

1. A request is received from an end-user's browser that, if fulfilled, would lead to an action requiring authorization taking place.

2. If the end-user is already authenticated:

- If the end-user is allowed to perform the action, the action proceeds
- If the end-user is NOT allowed to perform the action, an authorization error is displayed.

3. If the end-user is NOT authenticated, i.e. is accessing DSpace anonymously:

- The parameters etc. of the request are stored
- The startAuthentication method is invoked on the currently configured SiteAuthenticator implementation
- That startAuthentication might instantly authenticate the user somehow, or forward the request to some sort of log-in page; the parameters of the original request are safely stored and will be accessible after the log-in process has completed
- If authentication is successful, the original request is resumed from step 2. above.

Please see the SiteAuthenticator.java source file for information about each of the methods. The default implementation, org.dspace.app.webui.SimpleAuthenticator, is a simple implementation that implements these policies:

- Use of inbuilt e-mail address/password-based log-in. This is achieved by forwarding a request that is attempting an action requiring authorization to the password log-in servlet, /password-login. The password log-in servlet (org.dspace.app.webui.servlet.PasswordServlet contains code that will resume the original request if authentication is successful, as per step 3. described above.

- Users can register themselves (i.e. add themselves as e-people without needing approval from the administrators), and can set their own passwords when they do this
- Users are not members of any special (dynamic) e-person groups

Included in the source is the implementation of SiteAuthenticator we use here at MIT, edu.mit.dspace.MITAuthenticator. This implements a slightly more complex authentication mechanism:

- If an authentication user in an MIT user, they must log in using an X509 Web certificate. The certificate-login servlet, similar to the password-login servlet, authenticates users via these certificates, and if successful, resumes the original request just as the password log-in servlet would.

- MIT users are also automatically added to the special (dynamic) group called 'MIT Users' (which must be present in the system!). This allows us to create authorization policies for MIT users without having to manually maintain membership of the MIT users group.

- Anyone can register themselves, but MIT users doing this cannot set a password; they must use their X509 Web certificate to log in.

The X509 certificate login servlet has an extra feature: If the webui.cert.autoregister configuration property is true, it will automatically register the user with the system.

You could create a customized version of the password login servlet to perform a similar action. For example, if your organization uses Windows NT domain authentication, you could implement a version of PasswordServlet.java that validates against Windows NT authentication, and automatically adds an e-person record for new users. It is strongly recommended that you do not edit PasswordServlet but create a new servlet for this, so that future updates of the DSpace code do not overwrite your changes. You would also have to implement a customized SiteAuthenticator in which the startAuthentication method would forward requests to your new servlet.