Installing DSpace

For the Impatient

Since some users might want to get their test version up and running as fast as possible, offered below is an *unsupported* outline of getting DSpace to run quickly in a Unix-based environment using the DSpace source release.

> Only experienced unix admins should even attempt the following without going to the detailed Installation Instructions

```
useradd -m dspace
gzip xzf dspace-6.x-src-release.tar.gz
createuser --username=postgres --no-superuser --pwprompt dspace
createdb --username=postgres --owner=dspace --encoding=UNICODE dspace
psql --username=postgres dspace -c "CREATE EXTENSION pgcrypto;"
cd [dspace-source]/dspace/config/
cp local.cfg.EXAMPLE local.cfg
vi local.cfg
mkdir [dspace]
chown dspace [dspace]
su - dspace
cd [dspace-source]
mvn package
cd [dspace-source]/dspace/target/dspace-installer
ant fresh_install
cp -r [dspace]/webapps/* [tomcat]/webapps
/etc/init.d/tomcat start
[dspace]/bin/dspace create-administrator
```

Hardware Recommendations

You can install and run DSpace on most modern PC, laptop or server hardware. However, if you intend to run DSpace for a large community of potential end users, carefully review the Hardware Recommendations in the User FAQ

Prerequisite Software

The list below describes the third-party components and tools you'll need to run a DSpace server. These are just guidelines. Since DSpace is built on open source, standards-based tools, there are numerous other possibilities and setups.

Also, please note that the configuration and installation guidelines relating to a particular tool below are here for convenience. You should refer to the documentation for each individual component for complete and up-to-date details. Many of the tools are updated on a frequent basis, and the guidelines below may become out of date.

UNIX-like OS or Microsoft Windows

- UNIX-like OS (Linux, HP/UX, Mac OSX, etc.) : Many distributions of Linux/Unix come with some of the dependencies below pre-installed or easily installed via updates. You should consult your particular distribution's documentation or local system administrators to determine what is already available.
- Microsoft Windows: After verifying all prerequisites below, see the Windows Installation section for Windows tailored instructions

Java JDK 7 or 8 (OpenJDK or Oracle JDK)

OpenJDK download and installation instructions can be found here http://openjdk.java.net/install/. Most operating systems provide an easy path to install OpenJDK. Just be sure to install the full JDK (development kit), and not the JRE (which is often the default example).

Oracle's Java can be downloaded from the following location: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Make sure to download the appropriate version of the Java SE JDK.

Make sure to install the JDK and not just the JRE

At this time, DSpace requires the full JDK (Java Development Kit) be installed, rather than just the JRE (Java Runtime Environment). So, please be sure that you are installing the full JDK and not just the JRE.

Be aware that Tomcat 7 uses Java 1.6 to compile JSPs by default. See information about Tomcat below on how to configure it to use Java 1.7 for JSPs. Tomcat 8 uses Java 1.7 for JSPs by default. If you use another Servlet Container please refer to its documentation on this matter.

Optional ElasticSearch Usage Statistics feature has its own Java requirements

If you plan to use the (optional) Elasticsearch Usage Statistics feature in DSpace, the ElasticSearch backend provides its own recommendations regarding Java version.

http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/setup.html

"Elasticsearch is built using Java, and requires at least Java 7 in order to run. Only Oracle's Java and the OpenJDK are supported. We recommend installing the *Java 8 update 20 or later* , or *Java 7 update 55 or later* . Previous versions of Java 7 are known to have bugs that can cause index corruption and data loss."

(However, if you plan to use the Solr-based Usage Statistics that are enabled by default within DSpace, you can ignore these additional requirements.)

Apache Maven 3.0.5 or above (3.3.9+)* (Java build tool)

Maven is necessary in the first stage of the build process to assemble the installation package for your DSpace instance. It gives you the flexibility to customize DSpace using the existing Maven projects found in the *[dspace-source]/dspace/modules* directory or by adding in your own Maven project to build the installation package for DSpace, and apply any custom interface "overlay" changes.

If you will be building the Mirage 2 theme, you will need Maven 3.3.9 or above (see DS-2458 for details as to why).

Maven can be downloaded from the following location: http://maven.apache.org/download.html

Configuring a Proxy

You can configure a proxy to use for some or all of your HTTP requests in Maven. The username and password are only required if your proxy requires basic authentication (note that later releases may support storing your passwords in a secured keystore, in the mean time, please ensure your *settings.xml* file (usually *${user.home}/.m2/settings.xml*) is secured with permissions appropriate for your operating system).

Example:

```
<settings>
  .
  .
  <proxies>
   <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.somewhere.com</host>
      <port>8080</port>
      <username>proxyuser</username>
      <password>somepassword</password>
      <nonProxyHosts>www.google.com|*.somewhere.com</nonProxyHosts>
    </proxy>
  </proxies>
  .
  .
</settings>
```

Apache Ant 1.8 or later (Java build tool)

Apache Ant is required for the second stage of the build process (deploying/installing the application). First, Maven is used to construct the installer (`[dspace-source]/dspace/target/dspace-installer`), after which Ant is used to install/deploy DSpace to the installation directory.

Ant can be downloaded from the following location: http://ant.apache.org

Relational Database: (PostgreSQL or Oracle)

PostgreSQL 9.4 or later (with pgcrypto installed)

DSpace 6 requires Postgres 9.4+ with the pgcrypto extension enabled

PostgreSQL users MUST ensure they are running 9.4 or above AND have the pgcrypto extension installed and enabled.

The pgcrypto extension allows DSpace to create UUIDs (universally unique identifiers) for all objects in DSpace, which means that (internal) object identifiers are now globally unique and no longer tied to database sequences.

- PostgreSQL can be downloaded from http://www.postgresql.org/. It is also provided via many operating system package managers
  - If the version of Postgres provided by your package manager is outdated, you may wish to use one of the official PostgreSQL provided repositories:
    - Linux users can select their OS of choice for detailed instructions on using the official PostgreSQL apt or yum repository: http://www.postgresql.org/download/linux/
    - Windows users will need to use the windows installer: http://www.postgresql.org/download/windows/
    - Mac OSX users can choose their preferred installation method: http://www.postgresql.org/download/macosx/
- Install the pgcrypto extension. It will also need to be enabled on your DSpace Database (see Installation instructions below for more info).
  - On most Linux operating systems (Ubuntu, Debian, RedHat), this extension is provided in the "postgresql-contrib" package in your package manager. So, ensure you've installed "postgresql-contrib".
  - On Windows, this extension should be provided automatically by the installer (check your "[PostgreSQL]/share/extension" folder for files starting with "pgcrypto")
- Unicode (specifically UTF-8) support must be enabled (but this is enabled by default).
- Once installed, you need to enable TCP/IP connections (DSpace uses JDBC):
  - In `postgresql.conf`: uncomment the line starting: `listen_addresses = 'localhost'`. This is the default, in recent PostgreSQL releases, but you should at least check it.
  - Then tighten up security a bit by editing `pg_hba.conf` and adding this line: `host dspace dspace 127.0.0.1 255.255.255.255 md5`. This should appear *before* any lines matching `all` databases, because the first matching rule governs.
  - Then restart PostgreSQL.

Oracle 10g or later

- Details on acquiring Oracle can be downloaded from the following location: http://www.oracle.com/database/. You will need to create a database for DSpace. Make sure that the character set is one of the Unicode character sets. DSpace uses UTF-8 natively, and it is suggested that the Oracle database use the same character set. You will also need to create a user account for DSpace (e.g. *dspace*) and ensure that it has permissions to add and remove tables in the database. Refer to the Quick Installation for more details.
  - **NOTE:** If the database server is not on the same machine as DSpace, you must install the Oracle client to the DSpace server and point `tnsnames.ora` and `listener.ora` files to the database the Oracle server.
  - For people interested in switching from PostgreSQL to Oracle (or visa versa), you may be able to inves

Servlet Engine (Apache Tomcat 7 or later, Jetty, Caucho Resin or equivalent)

Tomcat 8 Version

Tomcat 8.0.32 (found e.g. in Debian 9 Stretch and Ubuntu 16.04 Xenial) has a bug which will cause PropertyBatchUpdateException or StringIndexOutOfBoundsException. This was fixed in 8.0.33. More information can be found in DS-3142.

Tomcat 7 Version

If you are using Tomcat 7, we recommend running Tomcat 7.0.30 or above. Tomcat 7.0.29 and lower versions suffer from a memory leak. As a result, those versions of tomcat require an unusual high amount of memory to run DSpace. This has been resolved as of Tomcat 7.0.30. More information can be found in DS-1553

- **Apache Tomcat 7 or later.** Tomcat can be downloaded from the following location: http://tomcat.apache.org.

- *The Tomcat owner* (i.e. the user that Tomcat runs as) ***must have read/write access*** *to the DSpace installation directory* (i.e. `[dspace]`). There are a few common ways this may be achieved:
    - One option is to specifically give the Tomcat user (often named "tomcat") ownership of the [dspace] directories, for example:

```
# Change [dspace] and all subfolders to be owned by "tomcat"
chown -R tomcat:tomcat [dspace]
```

    - Another option is to have Tomcat itself *run as* a new user named "dspace" (see installation instructions below). Some operating systems make modifying the Tomcat "run as" user easily modifiable via an environment variable named TOMCAT_USER. This option may be more desireable if you have multiple Tomcat instances running, and you do not want all of them to run under the same Tomcat owner.
- You need to ensure that Tomcat has a) enough memory to run DSpace and b) uses UTF-8 as its default file encoding for international character support. So ensure in your startup scripts (etc) that the following environment variable is set: *JAVA_OPTS ="-Xmx512M -Xms64M -Dfile.encoding=UTF-8"*
- **Modifications in *[tomcat]/conf/server.xml*** : You also need to alter Tomcat's default configuration to support searching and browsing of multi-byte UTF-8 correctly. You need to add a configuration option to the *<Connector>* element in *[tomcat]/config /server.xml*: *URIEncoding="UTF-8"* e.g. if you're using the default Tomcat config, it should read:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080"
           minSpareThreads="25"
           enableLookups="false"
           redirectPort="8443"
           connectionTimeout="20000"
           disableUploadTimeout="true"
           URIEncoding="UTF-8"/>
```

  You may change the port from 8080 by editing it in the file above, and by setting the variable *CONNECTOR_PORT* in *server.xml*. You should set the URIEncoding even if you are running Tomcat behind a proxy (Apache HTTPD, Nginx, etc.) via AJP.
- Tomcat 8 and above is using at least Java 1.7 for JSP compilation. However, by default, Tomcat 7 uses Java 1.6 for JSP compilation. If you want to use Java 1.7 in your .jsp files, you have to change the configuration of Tomcat 7. Edit the file called web.xml in the configuration directory of your Tomcat instance (${CATALINA_HOME}/conf in Tomcat notation). Look for a servlet definition using the org.apache.jasper.servlet.JSPServlet servlet-class and add two init parameters `compilerSourceVM` and `compilerTargetVM` as you see it in the example below. Then restart Tomcat.

${CATALINA_BASE}/conf/web.xml

```
 <servlet>
        <servlet-name>jsp</servlet-name>
        <servlet-class>org.apache.jasper.servlet.JspServlet<
/servlet-class>
        <init-param>
            <param-name>fork</param-name>
            <param-value>false</param-value>
        </init-param>
        <init-param>
            <param-name>xpoweredBy</param-name>
            <param-value>false</param-value>
        </init-param>
        <init-param>
            <param-name>compilerSourceVM</param-name>
            <param-value>1.7</param-value>
        </init-param>
        <init-param>
            <param-name>compilerTargetVM</param-name>
```

```
            <param-value>1.7</param-value>
        </init-param>
        <load-on-startup>3</load-on-startup>
    </servlet>
```

- **Jetty or Caucho Resin** DSpace will also run on an equivalent servlet Engine, such as Jetty (https://www.eclipse.org/jetty/) or Caucho Resin (http://www.caucho.com/). Jetty and Resin are configured for correct handling of UTF-8 by default.

Git (code version control)

Currently, there is a known bug in DSpace 6.x where a third-party Maven Module expects `git` to be available (in order to support the `./dspace version` commandline tool). We are working on a solution within this ticket: **DS-3418** - Getting issue details... STATUS

For the time being, you can work around this problem by installing Git locally: https://git-scm.com/downloads

Installation Instructions

Overview of Install Options

Two different distributions are available for DSpace, both of which require you to build the distribution using Apache Maven 3. The steps that are required to execute the build are identical. In a nutshell, the binary release build will download pre-compiled parts of DSpace, while the building the source release will compile most of DSpace's source code on your local machine.

It's important to notice that both releases will require outgoing internet connections on the machine or server where you are executing the build, because maven needs to download 3rd party dependencies that are not even included in the DSpace source release distribution.

- Binary Release (dspace-<version>-release.zip)
  - This distribution will be adequate for most cases of running a DSpace instance. It is intended to be the quickest way to get DSpace installed and running while still allowing for customization of the themes and branding of your DSpace instance.
  - This method allows you to customize DSpace configurations (in dspace.cfg) or user interfaces, using basic pre-built interface "overlays".
  - It downloads "precompiled" libraries for the core dspace-api, supporting servlets, taglibraries, aspects and themes for the dspace-xmlui, dspace-xmlui and other webservice/applications.
  - This approach only exposes selected parts of the application for customization. All other modules are downloaded from the 'Maven Central Repository' The directory structure for this release is the following:
    - *[dspace-source]*
      - *dspace/* - DSpace 'build' and configuration module
- Source Release (dspace-<version>-src-release.zip)
  - This method is recommended for those who wish to develop DSpace further or alter its underlying capabilities to a greater degree.
  - It contains **all** dspace code for the core dspace-api, supporting servlets, taglibraries, aspects and themes for Manakin (dspace-xmlui), and other webservice/applications.
  - Provides all the same capabilities as the binary release. The directory structure for this release is more detailed:
    - *[dspace-source]*
      - *dspace* - DSpace 'build' and configuration module
      - *dspace-api* - Java API source module
      - *dspace-jspui* - JSP-UI source module
      - *dspace-oai* - OAI-PMH source module
      - *dspace-rdf* - RDF source module
      - *dspace-rest* - REST API source module
      - *dspace-services* - Common Services module
      - *dspace-sword* - SWORD (Simple Web-serve Offering Repository Deposit) deposit service source module
      - *dspace-swordv2* - SWORDv2 source module
      - *dspace-xmlui* - XML-UI (Manakin) source module
      - *dspace-xmlui-mirage2* - Mirage 2 theme for the XMLUI
      - *pom.xml* - DSpace Parent Project definition

Overview of DSpace Directories

Before beginning an installation, it is important to get a general understanding of the DSpace directories and the names by which they are generally referred. (Please attempt to use these below directory names when asking for help on the DSpace Mailing Lists, as it will help everyone better understand what directory you may be referring to.)

DSpace uses three separate directory trees. Although you don't need to know all the details of them in order to install DSpace, you do need to know they exist and also know how they're referred to in this document:

1. **The installation directory**, referred to as `[dspace]`. This is the location where DSpace is installed and running. It is the location that is defined in the `dspace.cfg` as "dspace.dir". It is where all the DSpace configuration files, command line scripts, documentation and webapps will be installed.
2. **The source directory**, referred to as `[dspace-source]` . This is the location where the DSpace release distribution has been unpacked. It usually has the name of the archive that you expanded such as `dspace-<version>-release` or `dspace-<version>-src-release`. Normally it is the directory where all of your "build" commands will be run.
3. **The web deployment directory**. This is the directory that contains your DSpace web application(s).This corresponds to `[dspace]/webapps` by default. However, if you are using Tomcat, you may decide to copy your DSpace web applications from `[dspace]/webapps/` to `[tomcat]/webapps/` (with `[tomcat]` being wherever you installed Tomcat, also known as `$CATALINA_HOME`).
   For details on the contents of these separate directory trees, refer to directories.html. *Note that the* `[dspace-source]` *and* `[dspace]` *directories are always separate!*

If you ever notice that many files seems to have duplicates under `[dspace-source]/dspace/target` do not worry about it. This "target" directory will be used by Maven for the build process and you should not change any file in it unless you know exactly what you are doing.

Installation

This method gets you up and running with DSpace quickly and easily. It is identical in both the Default Release and Source Release distributions.

1. **Create the DSpace user *(optional)*** . As noted in the prerequisites above, Tomcat (or Jetty, etc) **must run as** an operating system user account that has full read/write access to the DSpace installation directory (i.e. `[dspace]`). Either you must ensure the Tomcat owner also owns `[dspace]`, OR you can create a new "dspace" user account, and ensure that Tomcat also runs as that account:

   ```
   useradd -m dspace
   ```

2. **Download** the latest DSpace release. There are two version available with each release of DSpace: (*dspace-n.x-release.* and *dspace-n.x-src-release.zip*); you only need to choose one. If you want a copy of all underlying Java source code, you should download the *dspace-n.x-src-release.zip.* Within each version, you have a choice of compressed file format. Choose the one that best fits your environment.
   1. Alternatively, you may choose to check out the latest release from the DSpace GitHub Repository. In this case, you'd be checking out the full Java source code. You'd also want to be sure to checkout the appropriate tag (e.g. dspace-6.0) or branch. For more information on using / developing from the GitHub Repository, see: Development with Git
3. **Unpack the DSpace software**. After downloading the software, based on the compression file format, choose one of the following methods to unpack your software:
   1. **Zip file**. If you downloaded *dspace-6.x-release.zip* do the following:

      ```
      unzip dspace-6.x-release.zip
      ```

   2. **.gz file**. If you downloaded *dspace-6.x-release.tar.gz* do the following:

      ```
      gunzip -c dspace-6.x-release.tar.gz | tar -xf -
      ```

   3. **.bz2 file**. If you downloaded _dspace-6.x-release.tar.bz do the following:

      ```
      bunzip2 dspace-6.x-release.tar.bz | tar -xf -
      ```

   For ease of reference, we will refer to the location of this unzipped version of the DSpace release as *[dspace-source]* in the remainder of these instructions. After unpacking the file, the user may wish to change the ownership of the *dspace-6.x-release* to the "dspace" user. (And you may need to change the group).
4. **Database Setup**
   - Also see "Relational Database" prerequisite notes above
   - *PostgreSQL:*
     - A PostgreSQL JDBC driver is configured as part of the default DSpace build. You no longer need to copy any PostgreSQL jars to get PostgreSQL installed.
     - Create a `dspace` database user (this user can have any name, but we'll assume you name them "dspace"). This is entirely separate from the `dspace` operating-system user created above:

       ```
       createuser --username=postgres --no-superuser --pwprompt
       dspace
       ```

You will be prompted (twice) for a password for the new `dspace` user. Then you'll be prompted for the password of the PostgreSQL superuser (`postgres`).

- Create a `dspace` database, owned by the `dspace` PostgreSQL user. Similar to the previous step, this can only be done by a "superuser" account in PostgreSQL (e.g. `postgres`):

```
createdb --username=postgres --owner=dspace --
encoding=UNICODE dspace
```

You will be prompted for the password of the PostgreSQL superuser (`postgres`).

- Finally, you MUST enable the pgcrypto extension on your new dspace database. Again, this can only be enabled by a "superuser" account (e.g. `postgres`)

```
# Login to the database as a superuser, and enable the
pgcrypto extension on this database
psql --username=postgres dspace -c "CREATE EXTENSION
pgcrypto;"
```

The "CREATE EXTENSION" command should return with no result if it succeeds. If it fails or throws an error, it is likely you are missing the required pgcrypto extension (see Database Prerequisites above).

- ***Alternative method: How to enable pgcrypto via a separate database schema.*** While the above method of enabling pgcrypto is perfectly fine for the majority of users, there may be some scenarios where a database administrator would prefer to install extensions into a database schema that is *separate from* the DSpace tables. Developers also may wish to install pgcrypto into a separate schema if they plan to "clean" (recreate) their development database frequently. Keeping extensions in a separate schema from the DSpace tables will ensure developers would NOT have to continually re-enable the extension each time you run a "`./dspace database clean`". If you wish to install pgcrypto in a separate schema here's how to do that:

```
# Login to the database as a superuser
psql --username=postgres dspace
# Create a new schema in this database named
"extensions" (or whatever you want to name it)
CREATE SCHEMA extensions;
# Enable this extension in this new schema
CREATE EXTENSION pgcrypto SCHEMA extensions;
# Grant rights to call functions in the extensions
schema to your dspace user
GRANT USAGE ON SCHEMA extensions TO dspace;


# Append "extensions" on the current session's
"search_path" (if it doesn't already exist in
search_path)
# The "search_path" config is the list of schemas that
Postgres will use
SELECT set_config('search_path',current_setting
('search_path') || ',extensions',false) WHERE
current_setting('search_path') !~ '(^|,)extensions(,
|$)';
# Verify the current session's "search_path" and make
sure it's correct
SHOW search_path;
```

```
# Now, update the "dspace" Database to use the same
"search_path" (for all future sessions) as we've set
for this current session (i.e. via set_config() above)
ALTER DATABASE dspace SET search_path FROM CURRENT;
```

- *Oracle:*
  - Setting up DSpace to use Oracle is a bit different now. You will need still need to get a copy of the Oracle JDBC driver, but instead of copying it into a lib directory you will need to install it into your local Maven repository. (You'll need to download it first from this location: http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html.) Run the following command (all on one line):

```
mvn install:install-file
    -Dfile=ojdbc6.jar
    -DgroupId=com.oracle
    -DartifactId=ojdbc6
    -Dversion=11.2.0.4.0
    -Dpackaging=jar
    -DgeneratePom=true
```

  - You need to compile DSpace with an Oracle driver (ojdbc6.jar) corresponding to your Oracle version - update the version in *[dspace-source]/pom.xml* E.g.:

```
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>11.2.0.4.0</version>
</dependency>
```

  - Create a database for DSpace. Make sure that the character set is one of the Unicode character sets. DSpace uses UTF-8 natively, and it is required that the Oracle database use the same character set. Create a user account for DSpace (e.g. *dspace*) and ensure that it has permissions to add and remove tables in the database.
  - NOTE: You will need to ensure the proper `db.*` settings are specified in your `local.cfg` file (see next step), as the defaults for all of these settings assuming a PostgreSQL database backend.

```
db.url = jdbc:oracle:thin:@host:port/SID
# e.g. db.url = jdbc:oracle:thin:@//localhost:1521/xe
# NOTE: in db.url, SID is the SID of your database defined
in tnsnames.ora
# the default Oracle port is 1521
# You may also use a full SID definition, e.g.
# db.url = jdbc:oracle:thin:@(description=(address_list=
(address=(protocol=TCP)(host=localhost)(port=1521)))
(connect_data=(service_name=DSPACE)))

# Oracle driver and dialect
db.driver = oracle.jdbc.OracleDriver
db.dialect = org.hibernate.dialect.Oracle10gDialect

# Specify DB username, password and schema to use
db.username =
db.password =
db.schema = ${db.username}
```

```
# For Oracle, schema is equivalent to the username of your
database account,
# so this may be set to ${db.username} in most scenarios
```

- Later, during the Maven build step, don't forget to specify `mvn -Ddb.name=oracle package`

5. **Initial Configuration (local.cfg):** Create your own `[dspace-source]/dspace/config/local.cfg` configuration file (you may wish to simply copy the provided `[dspace-source]/dspace/config/local.cfg.EXAMPLE`). This local.cfg file can be used to store *any* configuration changes that you wish to make which are local to your installation (see local.cfg configuration file documentation). ANY setting may be copied into this local.cfg file from the dspace.cfg or any other *.cfg file in order to override the default setting (see note below). For the initial installation of DSpace, there are some key settings you'll likely want to override, those are provided in the `[dspace-source]/dspace/config/local.cfg.EXAMPLE`. (NOTE: Settings followed with an asterisk (*) are highly recommended, while all others are optional during initial installation and may be customized at a later time)
    - `dspace.dir*` - must be set to the *[dspace]* (installation) directory (*NOTE: On Windows be sure to use forward slashes for the directory path!* For example: "`C:/dspace`" is a valid path for Windows.)
    - `dspace.hostname` - fully-qualified domain name of web server (or "localhost" if you just want to run DSpace locally for now)
    - `dspace.baseUrl*` - complete URL of this server's DSpace home page (including port), but without any context eg. /xmlui, /oai, etc.
    - `dspace.name` - "Proper" name of your server, e.g. "My Digital Library".
    - `solr.server*` - complete URL of the Solr server. DSpace makes use of Solr for indexing purposes.
    - `default.language` - Default language for all metadata values (defaults to "en_US")
    - `db.url*` - The full JDBC URL to your database (examples are provided in the `local.cfg.EXAMPLE`)
    - `db.driver*` - Which database driver to use, based on whether you are using PostgreSQL or Oracle
    - `db.dialect*` - Which database dialect to use, based on whether you are using PostgreSQL or Oracle
    - `db.username*` - the database username used in the previous step.
    - `db.password*` - the database password used in the previous step.
    - `db.schema*` - the database scheme to use (examples are provided in the local.cfg.EXAMPLE)
    - `mail.server` - fully-qualified domain name of your outgoing mail server.
    - `mail.from.address` - the "From:" address to put on email sent by DSpace.
    - `mail.feedback.recipient` - mailbox for feedback mail.
    - `mail.admin` - mailbox for DSpace site administrator.
    - `mail.alert.recipient` - mailbox for server errors/alerts (not essential but very useful!)
    - `mail.registration.notify`- mailbox for emails when new users register (optional)

    Your local.cfg file can override ANY settings from other *.cfg files in DSpace

    The provided `local.cfg.EXAMPLE` only includes a small subset of the configuration settings available with DSpace. It provides a good starting point for your own `local.cfg` file.

    However, you should be aware that ANY configuration can now be copied into your `local.cfg` to override the default settings. This includes ANY of the settings/configurations in:

    - The primary dspace.cfg file (`[dspace]/config/dspace.cfg`)
    - Any of the module configuration files (`[dspace]/config/modules/*.cfg` files)

    Individual settings may also be commented out or removed in your `local.cfg`, in order to re-enable default settings.

    See the Configuration Reference section for more details.

6. **DSpace Directory:** Create the directory for the DSpace installation (i.e. `[dspace]`). As *root* (or a user with appropriate permissions), run:

```
mkdir [dspace]
chown dspace [dspace]
```

(Assuming the *dspace* UNIX username.)

7. **Build the Installation Package:** As the *dspace* UNIX user, generate the DSpace installation package.

```
cd [dspace-source]
mvn package
```

Building with Oracle Database Support

Without any extra arguments, the DSpace installation package is initialized for PostgreSQL. If you want to use Oracle instead, you should build the DSpace installation package as follows:
`mvn -Ddb.name=oracle package`

Enabling and building the Mirage 2 theme (for XMLUI)

Mirage 2 is a responsive theme for the XML User Interface, added as a new feature in DSpace 5. It has not yet replaced the Mirage 1 theme as the XMLUI default theme.
The Mirage 2 build requires **git** to be installed on your server. Install git before attempting the Mirage 2 build.

To enable Mirage 2, add the following to the `<themes>` section of `[dspace-source]/dspace/config/xmlui.xconf`, replacing the currently active theme:

```
<theme name="Mirage 2" regex=".*" path="Mirage2/" />
```

It is important to do this before executing the maven build.

Mirage 2 is not yet activated in the default "mvn package" build. To include it as part of the build, run:

mvn package -Dmirage2.on=true

The speed of this specific step of the build can be increased by installing local copies of the specific dependencies required for building Mirage 2. The Mirage 2 developer documentation provides detailed instructions for these installations. After the installation of these dependencies, you can choose to run:

```
mvn package -Dmirage2.on=true -Dmirage2.deps.included=false
```

Warning: The Mirage 2 build process should NOT be run as "root". It must be run as a non-root user. For more information see: Mirage 2 Common Build Issues

8. **Install DSpace:** As the *dspace* UNIX user, install DSpace to `[dspace]`:

```
cd [dspace-source]/dspace/target/dspace-installer
ant fresh_install
```

To see a complete list of build targets, run: `ant help` *The most likely thing to go wrong here is the test of your database connection. See the Common Problems Section below for more details*.

9. **Decide which DSpace Web Applications you want to install.** DSpace comes with a variety of web applications (in `[dspace]/webapps`), each of which provides a different "interface" to your DSpace. Which ones you install is up to you, but there are a few that we highly recommend (see below):
    1. "xmlui" = This is the XML-based User Interface (XMLUI), based on Apache Cocoon. It comes with a variety of out-of-the-box themes, including Mirage 1 (the default) and Mirage 2 (based on Bootstrap). *Between the "xmlui" and "jspui", you likely only need to choose one.*
    2. "jspui" = This is the JSP-based User Interface (JSPUI), which is based on Bootstrap. *Between the "xmlui" and "jspui", you likely only need to choose one.*
    3. "solr" (*required*) = This is Apache Solr web application, which is used by the "xmlui" and "jspui" (for search & browse functionality), as well as the OAI-PMH interface. **It must be installed in support of either UI.**
    4. "oai" = This is the DSpace OAI interface. It allows for Metadata and Bitstream (content-file) harvesting, supporting OAI-PMH (Protocol for Metadata Harvest) and OAI-ORE (Object Reuse and Exchange) protocols
    5. "rdf" = This is the DSpace RDF interface supporting Linked (Open) Data.
    6. "rest" = This is the DSpace REST API
    7. "sword" = This is the DSpace SWORDv1 interface. More info on SWORD protocol and its usage.
    8. "swordv2" = This is the DSpace SWORDv2 interface. More info on SWORD protocol and its usage.
10. **Deploy Web Applications:** Please note that in the first instance you should refer to the appropriate documentation for your Web Server of choice. The following instructions are meant as a handy guide. You have two choices or techniques for having Tomcat/Jetty/Resin serve up your web applications:

- *Technique A.* Tell your Tomcat/Jetty/Resin installation where to find your DSpace web application(s). As an example, in the directory `[tomcat]/conf/Catalina/localhost` you could add files similar to the following (but replace `[dspace]`with your installation location):

DEFINE A CONTEXT FOR DSpace XML User Interface: xmlui.xml

```
<?xml version='1.0'?>
<Context
        docBase="[dspace]/webapps/xmlui"
        reloadable="true"
        cachingAllowed="false"/>
```

DEFINE A CONTEXT PATH FOR DSpace JSP User Interface: jspui.xml

```
<?xml version='1.0'?>
<Context
        docBase="[dspace]/webapps/jspui"
        reloadable="true"
        cachingAllowed="false"/>
```

DEFINE A CONTEXT PATH FOR DSpace Solr index: solr.xml

```
<?xml version='1.0'?>
<Context
        docBase="[dspace]/webapps/solr"
        reloadable="true"
        cachingAllowed="false"/>
```

DEFINE A CONTEXT PATH FOR DSpace OAI User Interface: oai.xml

```
<?xml version='1.0'?>
<Context
        docBase="[dspace]/webapps/oai"
        reloadable="true"
        cachingAllowed="false"/>
```

DEFINE ADDITIONAL CONTEXT PATHS FOR OTHER DSPACE WEB APPLICATIONS (REST, SWORD, RDF, etc.): [app].xml

```
<?xml version='1.0'?>
<!-- CHANGE THE VALUE OF "[app]" FOR EACH APPLICATION YOU WISH
TO ADD -->
<Context
        docBase="[dspace]/webapps/[app]"
        reloadable="true"
        cachingAllowed="false"/>
```

The name of the file (not including the suffix ".xml") will be the name of the context, so for example `xmlui.xml` defines the context at `http://host:8080/xmlui`. To define the *root context* (`http://host:8080/`), name that context's file `ROOT.xml`.

> **Tomcat Context Settings in Production**
>
> The above Tomcat Context Settings show adding the following to each `<Context>` element:
>
> ```
> reloadable="true" cachingAllowed="false"
> ```
>
> These settings are extremely useful to have when you are first getting started with DSpace, as they let you tweak the DSpace XMLUI (XSLTs or CSS) or JSPUI (JSPs) and see your changes get automatically reloaded by Tomcat (without having to restart Tomcat). However, it is worth noting that the Apache Tomcat documentation recommends Production sites leave the default values in place (`reloadable="false" cachingAllowed="true"`), as allowing Tomcat to automatically reload all changes may result in "significant runtime overhead".
>
> It is entirely up to you whether to keep these Tomcat settings in place. We just recommend beginning with them, so that you can more easily customize your site without having to require a Tomcat restart. Smaller DSpace sites may not notice any performance issues with keeping these settings in place in Production. Larger DSpace sites may wish to ensure that Tomcat performance is more streamlined.

- *Technique B.* Simple and complete. You copy only (or all) of the DSpace Web application(s) you wish to use from the [dspace] /webapps directory to the appropriate directory in your Tomcat/Jetty/Resin installation. For example:
  `cp -R [dspace]/webapps/* [tomcat]/webapps*` (This will copy all the web applications to Tomcat).
  `cp -R [dspace]/webapps/jspui [tomcat]/webapps*` (This will copy only the jspui web application to Tomcat.)

  To define the *root context* (`http://host:8080/`), name that context's directory `ROOT`.

11. **Administrator Account:** Create an initial administrator account from the command line:

```
[dspace]/bin/dspace create-administrator
```

12. **Initial Startup!** Now the moment of truth! Start up (or restart) Tomcat/Jetty/Resin. Visit the base URL(s) of your server, depending on which DSpace web applications you want to use. You should see the DSpace home page. Congratulations! Base URLs of DSpace Web Applications:
    - *JSP User Interface* - (e.g.) `http://dspace.myu.edu:8080/jspui`
    - *XML User Interface* (aka. Manakin) - (e.g.) `http://dspace.myu.edu:8080/xmlui`
    - *OAI-PMH Interface* - (e.g.) `http://dspace.myu.edu:8080/oai/request?verb=Identify` (Should return an XML-based response)

In order to set up some communities and collections, you'll need to login as your DSpace Administrator (which you created with `create-administrator` above) and access the administration UI in either the JSP or XML user interface.

Advanced Installation

The above installation steps are sufficient to set up a test server to play around with, but there are a few other steps and options you should probably consider before deploying a DSpace production site.

'cron' jobs / scheduled tasks

A few DSpace features **require** that a script is run regularly (via cron, or similar):

- the e-mail subscription feature that alerts users of new items being deposited;
- the 'media filter' tool, that generates thumbnails of images and extracts the full-text of documents for indexing;
- the 'checksum checker' that tests the bitstreams in your repository for corruption;
- the sitemap generator, which enhances the ability of major search engines to index your content and make it findable;
- the curation system queueing feature, which allows administrators to "queue" tasks (to run at a later time) from the Admin UI;
- and Discovery (search & browse), OAI-PMH and Usage Statistics all receive performance benefits from regular re-optimization

For much more information on recommended scheduled tasks, please see Scheduled Tasks via Cron.

Multilingual Installation

In order to deploy a multilingual version of DSpace you have to configure two parameters in *[dspace-source]/dspace/config/local.cfg:*

- *default.locale*, e.g. `default.locale = en`
- *webui.supported locales*, e.g. `webui.supported.locales = en, de`

The Locales might have the form country, country_language, country_language_variant.

According to the languages you wish to support, you have to make sure that all the i18n related files are available. See the Configuring Multilingual Support section for the JSPUI or the Multilingual Support for XMLUI in the configuration documentation.

DSpace over HTTPS

If your DSpace is configured to have users login with a username and password (as opposed to, say, client Web certificates), then you should consider using HTTPS. Whenever a user logs in with the Web form (e.g. *dspace.myuni.edu/dspace/password-login*) their DSpace password is exposed in plain text on the network. This is a very serious security risk since network traffic monitoring is very common, especially at universities. If the risk seems minor, then consider that your DSpace administrators also login this way and they have ultimate control over the archive.

The solution is to use *HTTPS* (HTTP over SSL, i.e. Secure Socket Layer, an encrypted transport), which protects your passwords against being captured. You can configure DSpace to require SSL on all "authenticated" transactions so it only accepts passwords on SSL connections.

The following sections show how to set up the most commonly-used Java Servlet containers to support HTTP over SSL.

Enabling the HTTPS support in Tomcat itself (running on ports 8080 and 8443)

Loosely based on http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html.

1. **For Production use:** Follow this procedure to set up SSL on your server. Using a "real" server certificate ensures your users' browsers will accept it without complaints. In the examples below, *$CATALINA_BASE* is the directory under which your Tomcat is installed.
    1. Create a Java keystore for your server with the password *changeit*, and install your server certificate under the alias *"tomcat"*. This assumes the certificate was put in the file *server.pem*:

```
$JAVA_HOME/bin/keytool -import -noprompt -v -storepass changeit
        -keystore $CATALINA_BASE/conf/keystore -alias tomcat -
file
        myserver.pem
```

    2. Install the CA (Certifying Authority) certificate for the CA that granted your server cert, if necessary. This assumes the server CA certificate is in *ca.pem*:

```
$JAVA_HOME/bin/keytool -import -noprompt -storepass changeit
        -trustcacerts -keystore $CATALINA_BASE/conf/keystore -
alias ServerCA
        -file ca.pem
```

    3. Optional – ONLY if you need to accept client certificates for the X.509 certificate stackable authentication module See the configuration section for instructions on enabling the X.509 authentication method. Load the keystore with the CA (certifying authority) certificates for the authorities of any clients whose certificates you wish to accept. For example, assuming the client CA certificate is in *client1.pem*:

```
$JAVA_HOME/bin/keytool -import -noprompt -storepass changeit
        -trustcacerts -keystore $CATALINA_BASE/conf/keystore  -
alias client1
        -file client1.pem
```

    4. Now add another Connector tag to your *server.xml* Tomcat configuration file, like the example below. The parts affecting or specific to SSL are shown in bold. (You may wish to change some details such as the port, pathnames, and keystore password)

```
<Connector port="8443"
               URIEncoding="UTF-8"
```

```
               minSpareThreads="25"
               enableLookups="false"
               disableUploadTimeout="true"
               scheme="https" secure="true" sslProtocol="TLS"
               keystoreFile="conf/keystore" keystorePass="
changeit"
               clientAuth="true" - ONLY if using client X.509
certs for authentication!
               truststoreFile="conf/keystore" truststorePass="
changeit" />
```

Also, check that the default Connector is set up to redirect "secure" requests to the same port as your SSL connector, e.g.:

```
<Connector port="8080"
               minSpareThreads="25"
               enableLookups="false"
               redirectPort="8443" />
```

2. **Quick-and-dirty Procedure for Testing:** If you are just setting up a DSpace server for testing, or to experiment with HTTPS, then you don't need to get a real server certificate. You can create a "self-signed" certificate for testing; web browsers will issue warnings before accepting it, but they will function exactly the same after that as with a "real" certificate. In the examples below, *$CATALINA_BASE* is the directory under which your Tomcat is installed.
    1. Create a new key pair under the alias name *"tomcat"*. When generating your key, give the Distinguished Name fields the appropriate values for your server and institution. CN should be the fully-qualified domain name of your server host. Here is an example:

```
$JAVA_HOME/bin/keytool -genkey \
  -alias tomcat \
  -keyalg RSA \
  -keysize 1024 \
  -keystore $CATALINA_BASE/conf/keystore \
  -storepass changeit \
  -validity 365 \
  -dname 'CN=dspace.myuni.edu, OU=MIT Libraries, O=Massachusetts
Institute of Technology, L=Cambridge, S=MA, C=US'
```

    You should be prompted for a password to protect the private key.

    Since you now have a signed server certificate in your keystore you can, obviously, skip the next steps of installing a signed server certificate and the server CA's certificate.
    2. Optional – ONLY if you need to accept client certificates for the X.509 certificate stackable authentication module See the configuration section for instructions on enabling the X.509 authentication method. Load the keystore with the CA (certifying authority) certificates for the authorities of any clients whose certificates you wish to accept. For example, assuming the client CA certificate is in *client1.pem*:

```
$JAVA_HOME/bin/keytool -import -noprompt -storepass changeit \
  -trustcacerts -keystore $CATALINA_BASE/conf/keystore -alias
client1 \
  -file client1.pem
```

    3. Follow the procedure in the section above to add another Connector tag, for the HTTPS port, to your *server.xml* file.

Using SSL on Apache HTTPD in front of Tomcat (running on ports 80 and 443)

When using Apache 2.4.2 (and lower) in front of a DSpace webapp deployed in Tomcat, mod_proxy_ajp and possibly mod_proxy_http breaks the connection to the back end (Tomcat) prematurely leading to response mixups. This is reported as bug CVE-2012-3502 (http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3502) of Apache and fixed in Apache 2.4.3 (see http://www.apache.org/dist/httpd/CHANGES_2.4). The 2.2.x branch hasn't shown this problem only the 2.4.x branch has.

Before following these instructions, it's *HIGHLY recommended* to first get DSpace running in standalone Tomcat on port 8080. Once DSpace is running, you can use the below instructions to add Apache HTTP Server in front of Tomcat in order to allow DSpace to run on port 80 and optionally port 443 (for SSL).

One of the easiest routes to both running DSpace on standard ports (80 and 443) as well as using HTTPS is to install Apache HTTP Server as your primary HTTP server, and use it to forward requests to Tomcat.

1. Install Apache HTTP Server alongside your Tomcat instance
2. In your Tomcat's `server.xml`, ensure that the AJP Connector is UNCOMMENTED. Usually this runs on port 8009, but you can decide to change the port if you desire

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" />
```

3. Choose how you'd like to redirect requests from Apache HTTP Server to Tomcat. There are two primary options (mod_proxy OR mod_jk), just choose ONE. (NOTE: "mod_proxy" is often the easier of the two):
    1. *OPTION 1*: Use "mod_proxy" and "mod_proxy_ajp" Apache modules to redirect requests to Tomcat (via AJP Connector) - RECOMMENDED
        1. Install "mod_proxy" and "mod_proxy_ajp" (usually they are installed by default with Apache HTTP Server)
        2. Enable both "mod_proxy" and "mod_proxy_ajp" modules. How you do this is often based on your operating system
            1. In Debian / Ubuntu, there's an "a2enmod" command that enables Apache 2 modules. So, you can just run: `sudo a2enmod proxy proxy_ajp`
            2. In other operating systems, you may need to find the appropriate "LoadModule" configurations (in Apache HTTP Server's main config) and uncomment it. You'll then need to restart Apache HTTP Server
        3. Create a new Virtual Host in Apache HTTP Server to represent your DSpace site. Here's a basic example of a Virtual Host responding to any port 80 requests for "my.dspace.edu":

```
<VirtualHost *:80>
   # Obviously, replace the ServerName with your DSpace
site URL
   ServerName my.dspace.edu

   ## Apache HTTP Server Logging Settings - modify how you
see fit
   ErrorLog ${APACHE_LOG_DIR}/my.dspace.edu-error.log
   CustomLog ${APACHE_LOG_DIR}/my.dspace.edu-access.log
combined

   # Possible values include: debug, info, notice, warn,
error, crit, alert, emerg.
   LogLevel warn

   # There are many more configurations available for
Virtual Hosts,
   # see the documentation for more details
   # http://httpd.apache.org/docs/2.4/vhosts/
</VirtualHost>
```

        4. If you want your site to also respond to SSL requests, you'll need to install and enable "mod_ssl" and create a second Virtual Host to respond to port 443 requests. An example is provided below. But much more details are available in the Apache HTTP SSL Documentation and the mod_ssl documentation

```
<VirtualHost *:443>
   # Obviously, replace the ServerName with your DSpace
site URL
   ServerName my.dspace.edu

   # You can have SSL Apache logging settings here too (see
the port 80 example above)

   # Configure your SSL Certificate (you must create one,
obviously)
   # See the "keytool" instructions above for examples of
creating this certificate
   # There are also many good guides on the web for
generating SSL certificates for Apache
   SSLEngine on
   SSLCertificateChainFile /path/to/your/chainfile.crt
   SSLCertificateFile      /path/to/your/public-cert.crt
   SSLCertificateKeyFile   /path/to/your/private-key.key

   # More information on SSL configurations can be found in
the mod_ssl documentation
   # http://httpd.apache.org/docs/2.4/mod/mod_ssl.html
</VirtualHost>
```

Extra SSL Configurations for X.509 Client Certificates authentication

If you are using X.509 Client Certificates for authentication: add these configuration options to the appropriate *httpd* configuration file, e.g. *ssl.conf*, and be sure they are in force for the virtual host and namespace locations dedicated to DSpace:

```
## SSLVerifyClient can be "optional" or "require"
SSLVerifyClient optional
SSLVerifyDepth 10
SSLCACertificateFile /path/to/your/client-CA-certificate
SSLOptions StdEnvVars ExportCertData
```

5. In *each* of your Apache HTTP Virtual Hosts (see above), use "ProxyPass" configurations to configure the redirects from Apache HTTP Server to Apache Tomcat. The exact configurations depend on whether you want to redirect ALL requests to Tomcat, or just certain paths. Here's a basic example. But much more information and examples can be found in the mod_proxy documentation

```
# These are just examples. THEY LIKELY WILL NEED
MODIFICATION.
# Again, remember to add these to your EXISTING
<VirtualHost> settings

<VirtualHost>

  ... (existing settings) ...

  # If there's a single path you do NOT want redirected,
you can use ! to ignore it
  # In this case any requests to "/ignored_path" will be
handled by Apache HTTPD and NOT forwarded to Tomcat
```

```
  ProxyPass  /ignored_path  !

  # These configurations say: By default, redirect ALL
requests to port 8009
  # (The port MUST match the port of your Tomcat AJP
Connector. By default this usually is 8009)
  ProxyPass         /  ajp://localhost:8009/
  ProxyPassReverse  /  ajp://localhost:8009/

  # You may also wish to provide additional "mod_proxy"
configurations,
  # for more examples and details see the documentation at
  # http://httpd.apache.org/docs/2.4/mod/mod_proxy.html
</VirtualHost>
```

2. OPTION 2: Alternatively, use "mod_jk" Apache module to redirect requests to Tomcat (via AJP Connector). For information on configuring mod_jk, please see the Apache Tomcat Connector documentation (specifically the "How To" on using the Tomcat Connector with Apache HTTP Server). You may also refer to our wiki guide for installing DSpace with ModJk.

4. Finally, restart your Apache HTTP Server and test things out.
    1. If you hit any issues, it is recommended to search around for guides to running Apache HTTP Server and Apache Tomcat using either "mod_proxy" or "mod_jk". DSpace does *not* require any unique configurations with regards to this redirection from Apache to Tomcat. So, any guides that generally explain how to redirect requests from Apache to Tomcat should also work for DSpace.

The Handle Server

First a few facts to clear up some common misconceptions:

- You don't **have** to use CNRI's Handle system. At the moment, you need to change the code a little to use something else (e.g PURLs) but that should change soon.
- You'll notice that while you've been playing around with a test server, DSpace has apparently been creating handles for you looking like *hdl:123456789/24* and so forth. These aren't really Handles, since the global Handle system doesn't actually know about them, and lots of other DSpace test installs will have created the same IDs. They're only really Handles once you've registered a prefix with CNRI (see below) and have correctly set up the Handle server included in the DSpace distribution. This Handle server communicates with the rest of the global Handle infrastructure so that anyone that understands Handles can find the Handles your DSpace has created.
- If you want to use the Handle system, you'll need to set up a Handle server. One is included with DSpace. Note that this is not required in order to evaluate DSpace; you only need one if you are running a production service. You'll need to obtain a Handle prefix from the central CNRI Handle site.

A Handle server runs as a separate process that receives TCP requests from other Handle servers, and issues resolution requests to a global server or servers if a Handle entered locally does not correspond to some local content. The Handle protocol is based on TCP, so it will need to be installed on a server that can send and receive TCP on port 2641.

You can either use a Handle server running on the same machine as DSpace, or you can install it on a separate machine. Installing it on the same machine is a little bit easier. If you install it on a separate machine, you can use one Handle server for more than one DSpace installation.

To install your Handle resolver on the host where DSpace runs:

> We recommend configuring your Handle server **without a passphrase**, as the current DSpace `start-handle-server` scripts do not yet support startup with a passphrase.
>
> If you choose to set a passphrase, you may need to start the Handle Server via: `[dspace]\bin\dspace dsrun net.handle.server.Main [dspace]\handle-server`

1. To configure your DSpace installation to run the handle server, run the following command:

```
[dspace]/bin/dspace make-handle-config [dspace]/handle-server
```

Ensure that *[dspace]/handle-server* matches whatever you have in *dspace.cfg* for the *handle.dir* property.

1. If you are using Windows, the proper command is:

```
[dspace]/bin/dspace dsrun net.handle.server.SimpleSetup [dspace]
/handle-server
```

Ensure that *[dspace]/handle-server* matches whatever you have in *dspace.cfg* for the *handle.dir* property.

2. Edit the resulting *[dspace]/handle-server/config.dct* file to include the following lines in the *"server_config"* clause:

```
"storage_type" = "CUSTOM"
"storage_class" = "org.dspace.handle.HandlePlugin"
```

This tells the Handle server to get information about individual Handles from the DSpace code.

3. Once the configuration file has been generated, you will need to go to http://hdl.handle.net/4263537/5014 to upload the generated sitebndl.zip file. The upload page will ask you for your contact information. An administrator will then create the naming authority/prefix on the root service (known as the Global Handle Registry), and notify you when this has been completed. You will not be able to continue the handle server installation until you receive further information concerning your naming authority.
4. When CNRI has sent you your naming authority prefix, you will need to edit the *config.dct* file. The file will be found in */[dspace]/handle-server*. Look for *"300:0.NA/YOUR_NAMING_AUTHORITY"*. Replace *YOUR_NAMING_AUTHORITY* with the assigned naming authority prefix sent to you.
5. Now start your handle server (as the dspace user):

```
[dspace]/bin/start-handle-server
```

1. If you are using Windows, there is a corresponding 'start-handle-server.bat' script:

```
[dspace]/bin/start-handle-server.bat
```

Note that since the DSpace code manages individual Handles, administrative operations such as Handle creation and modification aren't supported by DSpace's Handle server.

To install a Handle resolver on a separate machine:

The Handle server you use must be dedicated to resolve Handles from DSpace. You cannot use a Handle server that is in use with other software already. You can use CNRI's Handle Software -- all you have to do is to add to it a plugin that is provided by DSpace. The following instructions were tested with CNRI's Handle software version 7.3.1. You can do the following steps on another machine than the machine DSpace runs on, but you have to copy some files from the machine on which DSpace is installed.

1. Download the CNRI Handle Software: http://www.handle.net/download.html. In the tarball you'll find an `INSTALL.txt` with installation instructions -- follow it.
2. After installing the CNRI Handle Software you should have two directories: once that contains the CNRI software and one that contains the configuration of you local Handle Server. For the rest of this instruction we assume that the directory containing the CNRI Software is `/hs/hsj-7.3.1` and the directory containing the configuration of your local server is `/hs/srv_1`. (We use the same paths here as CNRIs INSTALL.txt.)
3. Download the plugin from https://github.com/DSpace/Remote-Handle-Resolver/releases. Select a release. You can get the source and build it yourself, or just use the JAR file included in the release. In either case, once you have a `dspace-remote-handle-resolver-VERSION.jar`, copy it to the directory containing the CNRI software (`/hs/hsj-7.3.1/lib`).
4. Create the directory `/hs/srv_1/logs`.
5. Create the following two files in `/hs/srv_1`.

log4j-handle-plugin.properties

```
log4j.rootCategory=INFO, A1
log4j.appender.A1=org.apache.log4j.DailyRollingFileAppender
log4j.appender.A1.File=/hs/srv_1/logs/handle-plugin.log
log4j.appender.A1.DatePattern= '.' yyyy-MM-dd
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d %-5p %c @ %m%n
log4j.logger.org.apache.axis.handlers.http.HTTPAuthHandler=INFO
```

Change the path in the third line, if necessary.

handle-dspace-plugin.cfg

```
dspace.handle.endpoint1 = http://example.org/dspace/handleresolver
```

If you are using XMLUI take a look in `[dspace-install]/config/dspace.cfg`, change the URL above to the value of your `dspace.url` and add `/handleresolver` to the end of it. If you are using JSPUI take a look in `[dspace-install]/config/dspace.cfg`, change the URL above to the value of your dspace.url and add `/json/hdlresolver` to the end of it. If you run more than one DSpace Installation, you may add more DSpace Endpoints. Just increase the number at the end of the key for each: `endpoint2`, `endpoint3`....

6. Edit the file `/hs/srv_1/config.dct` to include the following lines in the "`server_config`" clause:

```
"storage_type" = "CUSTOM"
"storage_class" = "org.dspace.handle.MultiRemoteDSpaceRepositoryHandlePlugin"
```

7. Copy `/hs/hsj-7.3.1/bin/hdl-server` to `/hs/srv_1/start-hdl-server`.
8. Edit `/hs/srv_1/start-hdl-server`:
    1. Find the last line that begins with `HDLHOME=`
    2. Below that line add the following one: `HDLHOME="/hs/hsj-7.3.1/"`
    3. Find a line that contains `exec java ... net.handle.server.Main ...`
    4. Add "`-Dlog4j.configuration=file:///hs/srv_1/log4j-handle-plugin.properties -Ddspace.handle.plugin.configuration=/hs/srv_1/handle-dspace-plugin.cfg`" right in front of `net.handle.server.Main`.
9. If your handle server is running, stop it.
10. From now on you should start this handle server using `/hs/srv_1/start-hdl-server`

Please note: The Handle Server will only start if it is able to connect to at least one running DSpace Installation. It only resolves the handles of the DSpace Installations that were running when it was started.

Updating Existing Handle Prefixes

If you need to update the handle prefix on items created before the CNRI registration process you can run the *[dspace]/bin/dspace update-handle-prefix script*. You may need to do this if you loaded items prior to CNRI registration (e.g. setting up a demonstration system prior to migrating it to production). The script takes the current and new prefix as parameters. For example:

```
[dspace]/bin/dspace update-handle-prefix 123456789 1303
```

This script will change any handles currently assigned prefix 123456789 to prefix 1303, so for example handle 123456789/23 will be updated to 1303/23 in the database.

Google and HTML sitemaps

To aid web crawlers index the content within your repository, you can make use of sitemaps. There are currently two forms of sitemaps included in DSpace: Google sitemaps and HTML sitemaps.

Sitemaps allow DSpace to expose its content without the crawlers having to index every page. HTML sitemaps provide a list of all items, collections and communities in HTML format, whilst Google sitemaps provide the same information in gzipped XML format.

To generate the sitemaps, you need to run *[dspace]/bin/dspace generate-sitemaps* This creates the sitemaps in *[dspace]/sitemaps/*

The sitemaps can be accessed from the following URLs (DSpace demo site is provided as example):

- http://demo.dspace.org/xmlui/sitemap - Index sitemap
- http://demo.dspace.org/xmlui/sitemap?map=0 - First list of items (up to 50,000)
- http://demo.dspace.org/xmlui/sitemap?map=n - Subsequent lists of items (e.g. 50,0001 to 100,000) etc...
  HTML sitemaps follow the same procedure:
- http://demo.dspace.org/xmlui/htmlmap - Index HTML based sitemap
- etc...

When running *[dspace]/bin/dspace generate-sitemaps* the script informs Google that the sitemaps have been updated. For this update to register correctly, you must first register your Google sitemap index page (*/dspace/sitemap*) with Google at http://www.google.com/webmasters/sitemaps/. If your DSpace server requires the use of a HTTP proxy to connect to the Internet, ensure that you have set *http.proxy.host* and *http.proxy.port* in *[dspace]/config/dspace.cfg*

The URL for pinging Google, and in future, other search engines, is configured in *[dspace]/config/dspace.cfg* using the *sitemap.engineurls* setting where you can provide a comma-separated list of URLs to 'ping'.

You can generate the sitemaps automatically every day using an additional cron job:

```
# Generate sitemaps at 6:00 am local time each day
0 6 * * * [dspace]/bin/dspace generate-sitemaps
```

More information on why we **highly recommend** enabling sitemaps can be found at Search Engine Optimization (SEO).

Statistics

DSpace uses the Apache Solr application underlaying the statistics. There is no need to download any separate software. All the necessary software is included. To understand all of the configuration property keys, the user should refer to DSpace Statistic Configuration for detailed information.

External database connection pool

Before it builds a pool of database connections, DSpace always tries to look up an existing, pre-configured pool in a directory service (if such a service is provided). Many web application containers supply such a service and can be configured to provide the connection pool to DSpace. If DSpace does not find a pre-configured pool, each web application will fall back to creating its own pool using the settings in `local.cfg`.

There are some advantages to using an external database pool:

- You can share one pool among several of DSpace's web applications—or even all of them. This can help economize database connections when one application uses many and another few. For example, if XMLUI needs 30 connections to run well at your site under peak load and OAI-PMH needs 5, you could connect them both to a pool of 35 connections, instead of letting each take 30 for a total of 60.
- You can have different pool sizes for the web applications and the command line tools. For example, configure an external pool with generous settings for the web applications, and a much smaller pool for the command line applications in `local.cfg`. Note: the command line tools cannot use an externally configured pool, and always use the settings in `local.cfg` to build their own pool.
- External database pooling often allows for more granular configuration of pool parameters and can even provide better performance than DSpace's fallback pooling (see the Tomcat JDBC Connection Pool documentation for more information).

DSpace applications will specifically look for an object named `jdbc/dspace`. The name is not configurable, but is specified in `config/spring/api/core-hibernate.xml` if you must know. You must configure the name of the directory object provided to your web application context(s) to match this. See below for an example in Tomcat.

An example in Tomcat

First, you must make the JDBC driver for your database available to Tomcat. For example, the latest PostgreSQL JDBC driver can be downloaded from the PostgreSQL project website and placed in Tomcat's `lib` directory. The exact location of this directory varies depending on your operating system and Tomcat version, but on Ubuntu 16.04 with Tomcat 7 the location would be `/usr/share/tomcat7/lib`.

Then add a `<Resource>` in Tomcat's `server.xml` to define the pool. The pool `name` here is global and can be anything you want:

server.xml

```
   <GlobalNamingResources>
...
    <Resource
        name='jdbc/instance'
        description='Our DSpace DBMS connection pool'
        type='javax.sql.DataSource'
        auth='Container'
        username='USER'
        password='SECRET'
        driverClassName='org.postgresql.Driver'
        url='jdbc:postgresql://dbms.example.com:5432/dspace'
```

```
          initialSize='5'
          maxTotal='50'
          maxIdle='15'
          minIdle='5'
          maxWaitMillis='5000'
          />
...
  </GlobalNamingResources>
```

Then add a <ResourceLink> to each web application's context configuration. The `name` parameter here is local to the application context, and must be `jdbc/dspace`:

```
<Context
...
  <ResourceLink
    name='jdbc/dspace'
    global='jdbc/instance'
    type='javax.sql.DataSource'
   />
...
</Context>
```

Notice that the `global` parameter in the `ResourceLink` matches the `name` of the global `Resource`. See the JNDI Datasource HOW-TO for more information about this configuration.

Windows Installation

Essentially installing on Windows is the same as installing on Unix so please refer back to the main Installation Instructions section.

- Download the DSpace source from GitHub and unzip it (WinZip will do this)
- If you install PostgreSQL, it's recommended to select to install the pgAdmin III tool. It provides a nice User Interface for interacting with PostgreSQL databases.
- For all path separators use forward slashes (e.g. "/"). For example: "C:/dspace" is a valid Windows path. But, be warned that "C:\dspace" IS INVALID and will cause errors.

Checking Your Installation

The administrator needs to check the installation to make sure all components are working. Here is list of checks to be performed. In brackets after each item, it the associated component or components that might be the issue needing resolution.

- System is up and running. *User can see the DSpace home page. [Tomcat/Jetty, firewall, IP assignment, DNS]*
- Database is running and working correctly. *Attempt to create a user, community or collection. [PostgreSQL, Oracle]* Run the database connection testing command to see if other issues are being reported: *[dspace]/bin/dspace database test*
- Email subsystem is running. The user can issue this command to test the email system: *[dspace]/bin/dspace test-email* It attempts to send a test email to the email address that is set in dspace.cfg (mail.admin). If it fails, you will get messages informing you as to why, referring you to the DSpace documentation.

Known Bugs

In any software project of the scale of DSpace, there will be bugs. Sometimes, a stable version of DSpace includes known bugs. We do not always wait until every known bug is fixed before a release. If the software is sufficiently stable and an improvement on the previous release, and the bugs are minor and have known workarounds, we release it to enable the community to take advantage of those improvements.

The known bugs in a release are documented in the *KNOWN_BUGS* file in the source package.

Please see the DSpace bug tracker for further information on current bugs, and to find out if the bug has subsequently been fixed. This is also where you can report any further bugs you find.

Common Problems

In an ideal world everyone would follow the above steps and have a fully functioning DSpace. Of course, in the real world it doesn't always seem to work out that way. This section lists common problems that people encounter when installing DSpace, and likely causes and fixes. This is likely to grow over time as we learn about users' experiences.

Common Installation Issues

- **Database errors occur when you run** `ant fresh_install`: There are two common errors that occur.
  - If your error looks like this:

```
[java] 2004-03-25 15:17:07,730 INFO
            org.dspace.storage.rdbms.InitializeDatabase @
Initializing Database
[java] 2004-03-25 15:17:08,816 FATAL
            org.dspace.storage.rdbms.InitializeDatabase @ Caught
exception:
[java] org.postgresql.util.PSQLException: Connection refused.
Check
            that the hostname and port are correct and that the
postmaster is
            accepting TCP/IP connections.
[java]     at
            org.postgresql.jdbc1.AbstractJdbc1Connection.
openConnection(AbstractJd
bc1Connection.java:204)
[java]     at org.postgresql.Driver.connect(Driver.java:139)
```

    it usually means you haven't yet added the relevant configuration parameter to your PostgreSQL configuration (see above), or perhaps you haven't restarted PostgreSQL after making the change. Also, make sure that the *db.username* and *db.password* properties are correctly set in *[dspace]/config/dspace.cfg*. An easy way to check that your DB is working OK over TCP/IP is to try this on the command line:

```
psql -U dspace -W -h localhost
```

    Enter the *dspace* database password, and you should be dropped into the psql tool with a *dspace=>* prompt.
  - Another common error looks like this:

```
[java] 2004-03-25 16:37:16,757 INFO
            org.dspace.storage.rdbms.InitializeDatabase @
Initializing Database
[java] 2004-03-25 16:37:17,139 WARN
            org.dspace.storage.rdbms.DatabaseManager @ Exception
initializing DB
            pool
[java] java.lang.ClassNotFoundException: org.postgresql.Driver
[java]     at java.net.URLClassLoader$1.run(URLClassLoader.java:
198)
[java]     at java.security.AccessController.doPrivileged(Native
           Method)
[java]     at
```

```
            java.net.URLClassLoader.findClass(URLClassLoader.java:
186)
```

This means that the PostgreSQL JDBC driver is not present in *[dspace]/lib*. See above.

- **GeoLite2-City Database file fails to download or install, when you run** `ant fresh_install`: There are two common errors that may occur:
  - If your error looks like this:

```
[get] Error getting http://geolite.maxmind.com/download/geoip
/database/GeoLite2-City.tar.gz to /usr/local/dspace/config
/GeoLite2-City.tar.gz

BUILD FAILED
/dspace-release/dspace/target/dspace-installer/build.xml:931:
java.net.ConnectException: Connection timed out
```

    it means that you likely either (a) don't have an internet connection to download the necessary GeoLite Database file (used for DSpace Statistics), or (b) the GeoLite Database file's URL is no longer valid.
  - Another common message looks like this:

```
[echo] WARNING : FAILED TO DOWNLOAD GEOLITE DATABASE FILE
[echo]          (Used for DSpace Solr Usage Statistics)
```

    Again, this means the GeoLite Database file cannot be downloaded or is unavailable for some reason. You should be able to resolve this issue by following the "Manually Installing/Updating GeoLite Database File" instructions.

General DSpace Issues

- **Tomcat doesn't shut down**: If you're trying to tweak Tomcat's configuration but nothing seems to make a difference to the error you're seeing, you might find that Tomcat hasn't been shutting down properly, perhaps because it's waiting for a stale connection to close gracefully which won't happen.
  - To see if this is the case, try running: `ps -ef | grep java` and look for Tomcat's Java processes. If they stay around after running Tomcat's *shutdown.sh* script, trying running `kill` on them (or `kill -9` if necessary), then starting Tomcat again.
- **Database connections don't work, or accessing DSpace takes forever**: If you find that when you try to access a DSpace Web page and your browser sits there connecting, or if the database connections fail, you might find that a 'zombie' database connection is hanging around preventing normal operation.
  - To see if this is the case, try running: `ps -ef | grep postgres`
  - You might see some processes like this:

```
dspace 16325  1997  0  Feb 14  ?         0:00 postgres: dspace
dspace    127.0.0.1 idle in transaction
```

    This is normal. DSpace maintains a 'pool' of open database connections, which are re-used to avoid the overhead of constantly opening and closing connections. If they're 'idle' it's OK; they're waiting to be used.
  - However sometimes, if something went wrong, they might be stuck in the middle of a query, which seems to prevent other connections from operating, e.g.:

```
dspace 16325  1997  0  Feb 14  ?         0:00 postgres: dspace
dspace    127.0.0.1 SELECT
```

    This means the connection is in the middle of a *SELECT* operation, and if you're not using DSpace right that instant, it's probably a 'zombie' connection. If this is the case, try running `kill` on the process, and stopping and restarting Tomcat.