Newtenberg Engine

DSpace 6.0

Autor: Bastián Martínez Henríquez

Índice

1. Introducción

Este documento tiene por itención servir como referencia para la instalación del software Dspace, enfocado en el uso e instalación para máquinas de la compañía Newtenberg. Además de esto. Posterior a esto se explica como utilizar DSpace en conjunto a los scripts creados para un proceso automático de creación y actualización de metadatos para los sitios publicados.

La estructura de la instalación funciona en dos máquinas distintas, una alojará a DSpace mientras que otra alojará los scripts que crean metadatos. Por lo tanto la primera se llamará máquina del engine y la segunda máquina del publicado.

La versión de DSpace para la cual se basa este documento es la 6.0 (Documentación oficial).

2. Requisitos de instalación

Para poder instalar DSpace se requieren las siguientes instalaciones:

2.1. Java JDK 7 o 8 (OpenJDK o Oracle JDK)

Para la versión de testing se utilizó OpenJDK, especificamente JDK 7.

2.2. Apache Maven 3.0.5 o mayor (3.3.9+)

Se utilizó la versión 3.5.2

2.3. Apache Ant 1.8 o mayor

Se utilizó la versión ya instalada en la máquina. Página oficial

2.4. Base Datos: PostgreSQL o Oracle

Se utilizó la versión 9.6.5. de PostgreSQL que se encontraba instalada en la máquina de Testing.Además se instaló la extensión pgcrypto. Por defecto estaba habilitado el soporte a UTF-8 que es necesario.

Debido a la estructura funcional del engine es necesario modificar la configuración de Postgres. El siguiente ejemplo es la cofiguración utilizada en testing:

```
local   all             postgres                                peer
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
#local   dspace          dspace                                  md5
local   all             all                                     trust
# IPv4 local connections:
#host    dspace          dspace          127.0.0.1/32            md5
host    all             all             127.0.0.1/32            password
# IPv6 local connections:
host    all             all             ::1/128                 password
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            md5
#host    replication     postgres        ::1/128                 md5
```

Figura 1: Configuración Postgres Pillan

2.5. Servlet Engine: Apache Tomcat, Jetty, o equivalente

La versión instalada de Tomcat en la máquina de testing es la 8.0.14. Para correr al menos los 4 app principales: XMLUI, REST, Solr y Oai se requiere aumentar la memoria PerGen de java (para testing se aumentó a 256Mb).

2.6. Git

Debido a un bug no solucionado de DSpace se requiere teneer instalado Git.

2.7. Distribución

DSpace se distribuye de dos formas distintas:

1. Binary Release
2. Source Release

Para la instalación en la máquina de testing se utilizó la versión Source, esta contiene todo el código de DSpace.

3. Instalación

La carpeta con la distribución de DSpace se llamará [dspace-source] mientras que la carpeta donde se instalará dspace se llamará [dspace]

3.1. Extracción de archivos

La versión de DSpace utilizada fue la 6.2 (Descarga en Github)
Se debe extraer el archivo dspace-6.x-src-release.zip. El directorio utilizado en testing es /opt/dspace-6.x-src-release

3.2. Configuración de Base de Datos

Para el caso de PostgreSQL, el driver JDBC viene configurado por defecto.
Se requiere además:

1. Crear un usuario: el usado en la máquina de testing se llama dspace:

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

2. Crear la Base de Datos:

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

3. Activar la extensión pgcrypo :

```
psql --username=postgres dspace -c "CREATE EXTENSION pgcrypto;"
```

3.3. Configuración de DSpace

En primer lugar se debe crear una archivo local.cfg ubicado en la carperta fuente de DSpace. Existe un archivo de ejemplo por lo que basta con editarlo:

```
[dspace-source]/dspace/config/local.cfg.EXAMPLE
```

Las modificaciones realizadas en la máquina de testing fueron las siguientes:

- dspace.hostname = kila-testing.newtenberg.com (se utilizó la url de la máquina)
- dspace.baseUrl = http://kila-testing.newtenberg.com:9588 (url de la máquina con su respectivo puerto)
- dspace.name = DSpace Newtenberg (nombre que tendrá el repositorio DSpace)
- oai.harvester.eperson = bastian.martinez@ing.uchile.cl (correo del administrado)

- solr.server = http://localhost:9588/solr (se agrega el puerto en que está corriendo tomcat)
- db.url = jdbc:postgresql://:5439/dspace (en el puerto que está corriendo Postgres)
- db.username = dspace
- db.password = dspace
- handle.prefix = 1234 (el por defecto no pasa testing oai)

Además de esto se requiere crear el directorio donde se alojará dspace, en la máquina de testing se utilizó /opt/dspace. Es importante dar los permisos adecuados para el usuario que correrá tomcat (en el caso de la máquina, el usuario engine).

3.4. Instalación de DSpace

3.4.1. Maven

Como el usuario UNIX que correrá tomcat, se debe crear el paquete de instalación

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

3.4.2. Ant

Posteriormente se debe instalar DSpace:

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

3.4.3. Elegir las aplicaciones a instalar

En este paso se deben copiar las app a la carpeta webapps de Tomcat. Las aplicaciones necesarias son las siguientes:

1. xmlui: administración de items/colecciones/comunidades.
2. solr: para el manejo de la base de datos.
3. oai: servidor del protocolo oai-pmh
4. rest: para la creación, modificación y eliminación de items

3.4.4. Crear administrador

Para poder utilizar la aplicación se requiere crear una cuenta de administrador. Se recomienda utilizar el mismo correo utilizado en local.cfg:

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

Luego de estos pasos se deben encontrar operativas todas las aplicaciones de DSpace.

3.5. Configuraciones iniciales

Después de crear el administración se requiere hacer ciertas configuraciones para el correcto funcionamiento.

3.5.1. Crear una comunidad y coleccion

Se debe utilizar la app XMLUI con los datos del administrador para crear una comunidad y una colección inicial. Además de esto se deben dar todas las autorizaciones al usuario que poblará la base de datos (en el caso de la máquina de testing, el mismo administrador).

Es recomendable guardar el código de la colección creada puesto que se utilizará posteriormente por medio de los Scripts de creación de metadatos.

3.5.2. Habilitar la API Rest

Para poder poblar de forma automática la base de datos se requiere habilitar la API Rest, para esto se debe acceder al archivo ubicado en webapps/rest/WEB-INF/web.xml y comentar el bloque

```
< security-constraint >.
```

3.5.3. Deshabilitar formatos de metadatos

Se deben deshabilitar la entrega de metadatos en ciertos formatos para pasar correctamente el validador de repositorios OAI-PMH.

El archivo a editar se encuentra en dspace/config/crosswalks/oai/xoai.xml, donde se deben comentar los siguientes metadatos ubicados entre el tag

```
<Context baseurl="request" name="Default Context">
```

- didl
- dim
- etdms
- uketd_dc
- junii2

4. Configuración de Script

Para la creación de metadatos desde el lado del publicado se requieren tres archivos de configuración: xmlDict.json, itemDict.json y pnid.json

4.0.1. xmlDict.json

Este archivo se utiliza para el script createXml.php, contiene toda la información para crear los archivos xml de metadatos a agregar.

Este es un archivo Json que tiene el siguiente formato:

- host: url a la Api Rest de la máquina que posee la información de los artículos
- cookie: cookie para la Api Rest
- htmlPath: arreglo con el formato [[ubicación de los artículos en publicado, numero de canal del engine]]
- domain: url del publicado
- channel: arreglo de canales en que se buscará la información

Ejemplo usado en la máquina de testing:

```
{
  "host":"https://kila-testing.newtenberg.com/mod/rest/cgi/xml.cgi/Article/"
      ,
  "cookie":"_uid=514; __digest=7172b86399a4666fd154988704b75583; __tracking=
      mod_oai",
  "htmlPath": [["../605/w3-article*", 501], ["../601/w3-article*", 503]],
  "domain": "webtest.newtenberg.com",
  "channel": [501, 503]
}
```

4.0.2. itemDict.json

Este archivo contiene toda la información necesaria para agregar los metadatos a la base de datos por medio del a API Rest DSpace.

Formato del archivo:

- host: url a la Api Rest DSpace de la máquina
- email: correo del usuario con privilegios para agregar items
- pass: contraseña

- collection: código de la colección a agregar (obtenible desde la misma Api Rest DSpace
- xmlPath: ubicación del txt que guarda los archivos xml a agregar

Ejemplo:

```
{"host":"http://kila-testing.newtenberg.com:9588/rest/",
  "email":"bastian.martinez@ing.uchile.cl",
  "pass":"**********",
  "collection": "f0eec72a-b3dc-44b8-b590-b689e2cdb260",
  "xmlPath": "xmlToAdd.txt" }
```

4.0.3. pnid.json

Este archivo contiene toda la información respecto a los nombres y valores de clasificando y su respectivo mapeo a formato dublin core.

Tiene el siguiente formato:

- numero de clasificando
- qname: mapeo dublin core
- filters: lista de filtros a aplicar

Ejemplo :

```
{
  "569":{"qname": "dc.subject", "filters": ["texto_previo","formato","
     software","protocolo"]},
  "567":{"qname": "dc.subject.classification", "filters": []},
  "574":{"qname": "dcterms.spatial","filters":["jerarquia_inversa_comas"]}
}
```

5. Ejecución del proceso

Para que se ejecute el proceso automáticamente se requiere en primer lugar tener todos los archivos necesarios instalados.

5.1. Máquina de publicado

La carpeta Script debe encontrar en la siguiente ubicación:

```
\public_html
        \canal_1
        \canal_2
        ...
        \canal_n
        \wiki
        \script
```

Mientras que su estructura interna es la siguiente:

```
\script
        xmlUtils.php
        itemUtils.php
        createXml.php
        addItems.php
        xmlDict.json
        itemDict.json
        pnid.json
        run.php
```

Posterior a la primera ejecución del script se agregará un archivo lock y al carpeta con archivos xml de metadatos.

5.2. Máquina de Engine

En la máquina de engine se debe ubicar el archivo manager.php en una ubicación pública. La úbicación utilizada en la máquina de testing es: https://kila-testing.newtenberg.com/mod/oai/manager.php

5.3. Automatización del proceso

Para que se ejecute cada actualización de forma automática se debe asigar una llamada al archivo run.php ubicado dentro del script públicado, y como argumento se debe usar start-oai-sync.

Ejemplo:

```
curl webtest.newtenberg.com/script/run.php?action=start-oai-sync
```