NVDCMS 2.0

Guide d'administration

NVDCMS 2.0: Guide d'administration

Table des matières

Chapitre 1. Introduction

1.1. Prérequis

Le système de gestion de contenu nécessite un certain nombre de prérequis pour pouvoir fonctionner dans sa version minimale.

Les logiciels à installer au préalable sont les suivants :

- Système d'exploitation Unix (Linux 2.4.18+ ou Solaris 8+ sont recommandés). Sur Unix, pour le traitement des images, un serveur XWindow est nécessaire dans le cas des JDK ne supportant pas le mode "java.awt.headless=true" (notamment les JDK de version inférieure à 1.4). Le CMS fonctionne également sur architecture Windows (mais limité au niveau du système de fichiers).
- JAVA JDK 1.3 (JDK IBM 1.3.1 Service Release 3 recommandé) ou 1.4.
- Serveur EJB CMP 2.0 (actuellement les serveurs Weblogic 6.1 SP2 et JBoss 3.0.X sont validés pour une bonne utilisation de la plate-forme).
- Base de données compatible SQL92 (actuellement seules les bases PostgreSQL 7.2+, HSQLDB 1.7 et Oracle 8i/9i sont validées pour une bonne utilisation de la plate-forme).

Les logiciels optionnels suivants sont nécessaires pour bénéficier de fonctionnalités suplémentaire :

- Serveur DNS pour la gestion des domaines (BIND8 minimum recommandé)
- Serveur Linux avec LVS pour le load-balancing des requêtes

1.2. Architecture générale

Les 3 éléments clé de l'architecture logicielle sont :

- Les serveurs Web avec le servlet Novadeck de gestion de contenu
- Le/les serveurs EJB
- La/les bases de données

Les serveurs web frontaux sont connectés aux serveurs EJB eux-même connectés aux bases de données. Les applications peuvent être distribuées sur différentes machines (qui communiquent à travers un réseau local privé) ou fonctionner sur une seule machine. Généralement un composant de load-balancing est placé en amont des serveurs web pour distribuer les requêtes réseaux. Les serveurs sont de préférence placés dans un réseau local privé sécurisé par une passerelle. Cependant pour certaines applications comme les WebServices ou la collecte d'informations distantes, il est nécessaire d'être en mesure d'effectuer des requêtes HTTP(S) vers l'extérieur. De plus, dans certains cas de figure, les serveurs web doivent être en mesure d'effectuer des requetes HTTP(S) sur eux même ce qui nécessite, soit la présence d'un DNS interne pour la résolution de nom dans des adresses IP privées, soit une configuration de la passerelle/load-balancer pouvant autoriser le load-balancing provenant du réseau privé à destination d'adresse publique devant être translatée à nouveau dans le réseau privé.

Novadeck Content Management System fonctionne avec n'importe quel serveur web supportant les Servlet Java 2.3+. Novadeck a sélectionné le serveur web Java Jetty pour sa stabilité et ses performances et l'a directement intégré au logiciel.

1.3. Compte utilisateurs Unix

Pour des raisons de sécurité, les différents logiciels doivent être installés, et exécutés avec des comptes Unix spécifiques.

Par défaut toutes les installations doivent être effectuées avec un compte administrateur (typiquement le compte 'admin').

Par défaut toutes les applications web doivent être exécutées avec un compte 'webuser' (typiquement le compte 'wwwrun').

Les autres applications comme les bases de données bénéficient généralement de leur propre compte (par exemple oracle ou postgres).

Chapitre 2. NVD-CMS

2.1. Installation

Le logiciel de gestion de contenu Novadeck CMS 2.0 doit être installé avec ses sous arborescences dans le répertoire Unix :

```
/usr/local/nvdcms/
```

2.2. Arborescence

Le répertoire nvdcms/ se décompose en deux sous-arborescences :

- ./lib : ce répertoire comprend l'ensemble des librairies utilisées par l'application
- ./etc : regroupe l'ensemble des fichiers de configuration XML de la plate-forme

Le répertoire ./etc regroupe les fichiers et dossiers suivants :

- jetty/ : répertoire contenant les fichiers de configuration du serveur httpd.
- jmx-xsl/ : répertoire contenant les feuilles de style du service JMX.
- licence.xml : fichier de licence.
- serverComponentConfig.xml : fichier de configuration des principaux paramètres de connexion du serveur.
- processesConfig.xml : configuration des processus d'exécution des documents.
- commandsConfig.xml : configuration des commandes exécutées sur la plate-forme.
- cachesConfig.xml : configuration des différents caches.
- componentConfig.xml : déclaration des entities.
- miscComponentConfig.xml : configuration des divers composants.
- logkit.xml : configuration des différents log.
- roleConfig.xml : définit des rôles pour améliorer la lisibilité des fichiers de configuration.

2.3. Configuration

La configuration du CMS se décompose en deux parties : la configuration des différents composants du serveur et la configuration httpd (voir Section 3.2).

La configuration des composants du serveur concerne essentiellement les fichiers suivants : la déclaration des 'entities' en tête du fichier "componentConfig.xml" et les différents paramètres de connexion dans "serverComponentConfig.xml".

2.3.1. Configuration du fichier "componentConfig.xml"

Le fichier "componentConfig.xml" inclut tous les autres fichiers de configartion. En tête du fichier, sont regroupées les déclaration des 'entities' XML.

```
<!ENTITY user-login-regexp '^[a-z0-9_\\.]{1,15}$'>
```

```
<!ENTITY user-password-regexp '^[A-Za-z0-9]{5,16}$'>
<!ENTITY user-email-regexp '^[A-Za-z0-9_\\.\\-]+@[a-z0-9_\\.\\-]+'>

<!ENTITY site-name-regexp '^[a-z0-9]{2,64}$'>
<!ENTITY site-description-regexp
'^[A-Za-z 0-9&apos;&amp;çéèêëàâäãôöõûüùîï;,:!?\\.\\-_]{0,128}$'>

<!ENTITY domain-name-regexp '^[a-z0-9\\.\\-]{3,100}$'>
<!ENTITY parked-domain-name-regexp '^[a-z0-9\\.\\-]{3,100}$'>

<!ENTITY common-name-regexp '^[a-z0-9\\-]{3,32}$'>


<!ENTITY sso-host 'sso.nvdcms.org'>
<!ENTITY jndi-provider 'jboss'>
<!ENTITY nvdcms-home '/usr/local/nvdcms'>
```

La signification des différents 'entities' est la suivante :

- user-password-regexp : expression régulière qui caractèrise la syntaxe d'un mot de passe utilisateur.
- user-email-regexp : expression régulière qui caractèrise la syntaxe d'un email d'un utilisateur.
- site-name-regexp : expression régulière qui caractèrise la syntaxe d'un nom de sous domaine de site.
- site-description-regexp : expression régulière qui caractèrise la syntaxe d'une description de site.
- domain-name-regexp : expression régulière qui caractèrise la syntaxe d'un nom de domaine.
- parked-domain-name-regexp : expression régulière qui caractèrise la syntaxe d'un domaine parqué.
- common-name-regexp : expression régulière qui caractérise la syntaxe des différents noms utilisés dans la plate-forme (nom d'objet, d'une liste de publication, etc...).
- sso-host : nom du site qui gère le Single Sign On

2.3.2. Configuration du fichier "serverComponentConfig.xml"

Les sections "key-store", "license-checker", "signature-helper" et "signature-generator" sont utilisées pour la gestion des clés privées et certificats. L'emplacement du 'keystore' est indiqué dans l'élément "store-file".

```
<key-store logger="license.key-store">
<type>JKS</type>
<store-file>&nvdcms-home;/etc/keystore-lite.jks</store-file>
<password>stouen</password>
</key-store>

<license-checker logger="license.license-checker">
<licence-file-path>&nvdcms-home;/etc/licence.xml</licence-file-path>
<certificate-alias>NVDCMS-Licence</certificate-alias>
</license-checker>

<signature-helper logger="license.signature-helper"/>

<signature-generator logger="license.signature-generator">
<algorithm>SHA1withDSA</algorithm>
</signature-generator>
```

La section "jndi-context-providers" permet de configurer le serveur JNDI et qui permettra par extension de se connecter au serveur EJB. Ici nous configurons deux composants pour la connexion au serveur JNDI (WebLogic et Jboss) via "java.naming.provider.url".

```
<jndi-context-providers logger="setup.JNDI.contextProviderSelector">
```

```
<initial-context name="jboss" logger="tools.JNDI.jboss">
<environment key="java.naming.provider.url"
>jnp://10.0.1.95:1099</environment>
<environment key="java.naming.security.principal">login</environment>
<environment key="java.naming.security.credentials">pass</environment>
<environment key="java.naming.factory.initial"
>org.jnp.interfaces.NamingContextFactory</environment>
</initial-context>

<initial-context name="weblogic" logger="tools.JNDI.weblogic">
<environment key="java.naming.provider.url"
>t3://10.0.100.1:7001</environment>
<environment key="java.naming.security.principal">login</environment>
<environment key="java.naming.security.credentials">pass</environment>
<environment key="java.naming.factory.initial"
>weblogic.jndi.WLInitialContextFactory</environment>
</initial-context>
</jndi-context-providers>
```

La section "directory-context-providers" configure le composant qui assure la connexion au serveur LDAP.

```
<directory-context-providers
logger="setup.JNDI.directory.contextProviderSelector">
<initial-directory-context name="ldap" logger="tools.JNDI.directory.ldap">
<environment key="java.naming.provider.url"
>ldap://10.0.1.99:389/o=newsdeck</environment>
<environment key="java.naming.security.principal"
>cn=admin, o=newsdeck</environment>
<environment key="java.naming.security.credentials"
>pass</environment>
<environment key="java.naming.factory.initial"
>com.sun.jndi.ldap.LdapCtxFactory</environment>
<environment key="java.naming.ldap.version">2</environment>
<environment key="java.naming.security.authentication"
>simple</environment>
</initial-directory-context>
</directory-context-providers>
```

La section "ejb" configure le composant qui assure l'initialisation de la connexion au serveur EJB ainsi que la configuration des composants logiciel de la plate-forme contenu dans le serveur EJB.

L'élément "jndi-provider" indique le serveur JNDI à utiliser (et par extension le serveur EJB). L'élément "database" comprend les descriptifs de la connexion du serveur EJB à la base de données. L'élément "search-engine" précise les différents paramètre à passer au composant qui gére le moteur de recherche sur le serveur. L'élément "query-cache-size" détaille la taille du cache de requête SQL sur le serveur EJB. L'élément "publication-daemon-idle-time" indique la durée de sommeil en secondes du 'daemon' interne qui gère la publication/dépublication.

```
<ejb logger="tools.EJB">

<jndi-provider>&jndi-provider;</jndi-provider>

<database>
<!-- for HSQL database
<adaptor>org.nvdcms.cms.ejb.sql.adaptor.HSQLDBAdaptor</adaptor>

for Oracle database
<adaptor>org.nvdcms.cms.ejb.sql.adaptor.OracleDBAdaptor</adaptor>
-->
<!-- for Postgresql database
-->
<adaptor>org.nvdcms.cms.ejb.sql.adaptor.PGDBAdaptor</adaptor>

<!-- -->
<!-- for jboss -->
```

```
<logical-name>java:/jdbcPool1</logical-name>
<!-- for weblogic
<logical-name>java:comp/env/jdbcPool1</logical-name>
-->
</database>

<search-engine>
<max-keywords>4</max-keywords>
<word-min-size>3</word-min-size>
<word-max-size>32</word-max-size>
<word-cache-size>20000</word-cache-size>
</search-engine>

<query-cache-size>200</query-cache-size>

<!-- Time in second -->
<publication-daemon-idle-time>60</publication-daemon-idle-time>

</ejb>
```

La section "jms-distributed-map" défini le serveur JMS. L'élément "jndi-provider" indique le serveur JNDI à utiliser (et par extension le serveur JMS). L'élément "distributed-map-topic-name" indique le nom du 'topic' pour les map distribuées qui doit également être défini avec le même nom dans le serveur JMS.

```
<jms-distributed-map logger="jms.distributedmap">
<jndi-provider>&jndi-provider;</jndi-provider>
<jms>
<factory>javax.jms.TopicConnectionFactory</factory>
<!-- !NOTICE! The topic name MUST be declared in JMS server -->
<distributed-map-topic-name
>topic/DISTRIBUTED_MAP</distributed-map-topic-name>
<default-timeout>2000</default-timeout>
</jms>
</jms-distributed-map>
```

La section "user-transaction-factory" défini le serveur EJB qui gère les transactions.

```
<user-transaction-factory logger="tools.Transaction">
<jndi-provider>&jndi-provider;</jndi-provider>
<jndi-name>UserTransaction</jndi-name>
</user-transaction-factory>
```

La section "org.nvdcms.cms.mail.MailSender" caractèrise le composant qui assure l'envoi des emails. L'élément "default-server" indique l'adresse par défaut du serveur SMTP. L'élément "default-charset" indique le charset à utiliser par défaut. L'élément "postmaster" indique l'adresse email que la plate-forme doit utiliser pour l'envoi de messages internes.

```
<component class="org.nvdcms.cms.mail.MailSender"
role="org.nvdcms.cms.mail.MailSender"
logger="tools.MailSender">
<default-server>10.0.1.92</default-server>
<default-charset>ISO-8859-1</default-charset>
<postmaster>nvd-cms@nvdcms.org</postmaster>
</component>
```

La section "org.nvdcms.cms.tools.CronExecutor" assure la configuration du composant chargé d'exécuter les tâche periodiques. Les principaux éléments à configurer sont : l'élément "queue-name" qui indique le nom de la 'queue' JMS à utiliser, celle-ci devant être également définie sous le même nom dans le serveur JMS et l'élement "cron-daemon-idle-time" qui indique la période de sommeil du 'daemon' interne chargé de déclancher les tâches.

```
<component class="org.nvdcms.cms.tools.CronExecutor"
role="org.nvdcms.cms.tools.CronExecutor"
logger="tools.CronExecutor">
<jndi-provider>&jndi-provider;</jndi-provider>
```

```
<jms>
<queue-name>queue/NVDCRON</queue-name>
<queue-factory-name>javax.jms.QueueConnectionFactory</queue-factory-name>
<timeout>10000</timeout>
</jms>

<!-- in seconds -->
<cron-daemon-idle-time>300</cron-daemon-idle-time>

<dom-loader>CachedLoader</dom-loader>
<document-executor>default</document-executor>
</component>
```

La section "org.nvdcms.cms.execution.module.nsp.NspTools" et "org.nvdcms.cms.execution.module.nsp.SimpleDomGeneratorFactory" définisse la configuration des composants qui prennent en charge les scripts NSP. L'élément "classpaths" indique les différents 'classpath' autorisés pour la compilation et l'exécution des scripts. L'élément "allowed-classname" liste les différentes classes pouvant être utilisées dans les scripts NSP. L'élément "compile-folder" précise le sous répertoire de compilation, relatif au répertoire temporaire de la plate-forme.

```
<component class="org.nvdcms.cms.execution.module.nsp.NspTools"
role="org.nvdcms.cms.execution.module.nsp.NspTools"
logger="tools.NSP">
<classpaths>
<classpath>&nvdcms-home;/lib/nvdcms.jar</classpath>
<classpath>&nvdcms-home;/lib/xml/jaxp.jar</classpath>
<classpath>&nvdcms-home;/lib/xml/xercesImpl.jar</classpath>
<classpath>&nvdcms-home;/lib/xml/xmlParserAPI.jar</classpath>
</classpaths>
<allowed-classname>
<class>java.lang.Long</class>
<class>java.lang.String</class>
</allowed-classname>
</component>

<component
class="org.nvdcms.cms.execution.module.nsp.SimpleDomGeneratorFactory"
role="org.nvdcms.cms.execution.module.nsp.SimpleDomGeneratorFactory"
logger="tools.NSP">
<!-- relative folder in temp directory -->
<compile-folder>/nvd.nsp</compile-folder>
<transformer-factory>fragment</transformer-factory>
</component>
```

La section "file-access-manager" définit les différents répertoires utilisés par la plate-forme. L'élément "file-root" indique l'emplacement dans lequel seront placés tous les fichiers de tous les sites de la plate-forme (ce répertoire est généralement partagé par toutes les machines du cluster). L'élément "site-temp-dir" mentionne l'emplacement du répertoire temporaire relatif au répertoire de base d'un site (généralement sous la forme "/www/domain.com/s/i/site/"). L'élément "local-temp-dir" comprend l'emplacement du répertoire temporaire non partagé (utilisé notamment pour le swap de donné et la compilation de script NSP). L'élément "shared-temp-dir" indique le répertoire temporaire commun à tous les serveurs d'un cluster de machine.

```
<file-access-manager class="org.nvdcms.cms.io.MultiSiteFileAccessManager"
logger="tools.fileManager">
<file-root>/www</file-root>

<site-temp-dir>/tmp</site-temp-dir>
<local-temp-dir>/tmp</local-temp-dir>
<shared-temp-dir>/tmp</shared-temp-dir>
</file-access-manager>
```

2.3.3. Gestion des polices de caractères de FOP

Il est possible d'ajouter des polices de caractères 'True Type' ou 'Adobe Type 1' à la liste des polices de caractères utilisées lors de la transformation d'un document XML en PDF (les polices de caractères seront jointes au PDF).

Il suffit de générer un fichier de 'metrics' à partir des fichiers de polices de caractères à l'aide des commandes suivantes (respectivement pour les TTF ou les Type1) :

```
export CLASSPATH=lib/render/fop.jar:lib/jakarta/avalon-framework.jar:\
lib/xml/xercesImpl.jar:lib/xml/xmlParserAPI.jar:\
lib/xml/jaxp.jar:lib/xalan.jar

java -Xmx64m -cp $CLASSPATH \
org.apache.fop.fonts.apps.TTFReader font.ttf metrics.xml


java -Xmx64m -cp $CLASSPATH \
org.apache.fop.fonts.apps.PFMReader font.pfm metrics.xml
```

Placer ensuite les fichiers de polices de caractères et de 'metrics' dans le répertoire /etc/fonts/ et éditer la section 'font' du fichier "fop-config.xml" en indiquant les noms des polices de caractères et le style/poids associés. Ces mêmes noms seront utilisés pour mentioner la police de caractère à utiliser dans les fichiers 'fo'.

```
<fonts>
<font metrics-file="/usr/local/nvdcms/etc/fonts/font.xml" kerning="yes"
embed-file="/usr/local/nvdcms/etc/fonts/font.ttf">
<font-triplet name="fontname" style="normal" weight="normal"/>
<font-triplet name="fontname" style="normal" weight="bold"/>
</font>
</fonts>
```

Dans cet exemple, la même font "font.ttf" est associée au nom 'fontname' style/poids 'normal/normal' et 'normal/bold'.

2.4. Gestion des clés de chiffrement du CMS

2.4.1. Générer un keystore avec clé publique/privée

Utiliser le programme "keytool" livré avec le JDK de la façon suivante :

```
> keytool -genkey -alias NVDCMS
-keyalg DSA
-dname "CN=NVD CMS, OU=NVD,O=Novadeck SA, C=FR"
-keypass keypassword1
-storepass storepassword
-keystore keystore.jks
-validity 3650
> keytool -genkey
-alias NVDCMS-Licence
-keyalg DSA -dname "CN=NVD CMS, OU=NVD,O=Novadeck SA, C=FR"
-keypass keypassword2
-storepass storepassword
-keystore keystore.jks
-validity 3650
```

L'alias et le storepass sont normalement définis en statique dans le code Licence.java. La "validity" est en nombre de jours.

2.4.2. Générer un certificat

Pour en extraire seulement la clé publique (certificat) et la placer dans un keystore

```
> keytool -export -alias nvdcms-licence
```

```
-storetype jks
-keystore keystore.jks
-file keystore-public.export
> keytool -import
-file keystore-public.export
-keystore keystore-public.jks
```

2.4.3. Signer un fichier

Utiliser le script shell :

```
licencegen.sh
```

indiquer le "licence-file" ainsi que "private keystore file" en paramètre de ligne de commande. Saisir la passphrase ("keypassword2" dans les exemples). Ajouter au document root ("licence") du fichier de licence.xml l'attribut généré.

2.5. JDK 1.3 et serveur X11

Sur unix et avec les JDK 1.3.x, les librairies de traitement d'images intégrés au CMS necessitent la presence d'un serveur X11.

Dans le cas d'une machine dédiée, il est possible d'utiliser le serveur X11 virtuel "Xvfb". L'installation de "Xvfb" nécessite deux packages (à installer à l'aide de la commande "rpm -ivh") :

```
xextra-*.i386.rpm
xfnt100-*.i386.rpm
```

Il faut alors lancer l'application et indiquer le DISPLAY (avec la commande bash export par exemple) à l'aide des commandes :

```
nohup Xvfb :0 -screen 0 640x480x16 &
export DISPLAY=127.0.0.1:0.0
```

Le JDK 1.4 dispose d'une option qui permet de ne pas avoir à utiliser un serveur X11. Il suffit de lancer l'application avec la propriété suivante :

```
-Djava.awt.headless=true
```

Chapitre 3. HTTPD

3.1. Arborescence

Les fichiers de configuration se trouve dans le repertoire etc/jetty/ sont les suivants :

- nvdcms.xml : fichier de configuration du serveur http (port, nombre de thread etc...).
- admin.xml : fichier de configuration du listener pour l'administration du serveur (qui assure l'arrêt et le redémarrage via le réseau).
- webdefault.xml : fichier de configuration du servlet NVDCMS (association de l'extension *.xml).

3.2. Configuration

Pour des raisons de sécurité le serveur httpd doit être exécuté avec le compte Unix 'wwwrun'. N'étant pas un compte root, le serveur doit être lancé avec port TCP d'écoute supérieur à 1024 (par défaut 8080) d'où la nécessité de configurer un règle de IPTable sous Linux si l'on veut renvoyer les connexions à destination du port 80 vers le port 8080.

3.2.1. Configuration du script shell "nvdcms.sh"

Positionnement des variables pour Jetty :

```
JAVA_HOME=/usr/local/jdk/
XM_OPTION="-Xms4m -Xmx16m"
JETTY_CONSOLE="/www/nvdcms2_"$HOSTNAME"_console"
```

- JAVA_HOME : emplacement de la JVM
- XM_OPTION : paramètre de mémoire de la JVM
- JETTY_CONSOLE : emplacement du fichier de sortie de Jetty

3.2.2. Configuration du fichier "nvdcms.xml"

Le fichier de configuration est le suivant :

```
<Configure class="org.nvdcms.cms.jetty.CmsServer">
<Call name="addListener">
<Arg>
<New class="org.mortbay.http.SocketListener">
<Set name="Port">8080</Set>
<Set name="MinThreads">1</Set>
<Set name="MaxThreads">25</Set>
<Set name="MaxStopTimeMs">5000</Set>
<Set name="MaxIdleTimeMs">10000</Set>
<Set name="LowResourcePersistTimeMs">5000</Set>
</New>
</Arg>
</Call>
<Call name="addWebApplication">
<Arg>/</Arg>
<Arg>/www</Arg>
<Arg><SystemProperty name="jetty.home"
default="."/>/etc/jetty/webdefault.xml</Arg>
<Arg type="boolean">false</Arg>
```

```
<!--
.htaccess apache style access authentification Configuration
-->
<Call name="addHandler">
<Arg><New class="org.mortbay.http.handler.NVDAccessHandler">
<Set name="AccessFile">.htaccess</Set></New>
</Arg>
</Call>
</Call>

</Configure>
```

La section "org.mortbay.http.SocketListener" indique la configuration de la socket d'écoute (ici sur le port 8080) et le nombre de threads qui sont générés pour la prise en compte des connexions (maximum 2000 et minimum 5).

La section :

```
<Call name="addWebApplication">
<Arg>/</Arg>
<Arg>/www</Arg>
<Arg><SystemProperty name="jetty.home"
default="."/>/etc/jetty/webdefault.xml</Arg>
<Arg type="boolean">false</Arg>
</Call>
```

Indique l'emplacement des répertoires racine de document du serveur web ainsi que les propriétés qui lui sont assignées. Ici, la racine du site se situe dans le répertoire "/www" et les propriétés qui lui sont assignées sont mentionnées dans le fichier "webdefault.xml" du répertoire de etc/jetty.

La section :

```
<Call name="addHandler">
<Arg><New class="org.mortbay.http.handler.NVDAccessHandler">
<Set name="AccessFile">.htaccess</Set></New>
</Arg>
</Call>
```

Autorise la prise en compte des fichiers .htaccess (qui permettent de restreindre l'accès à des répertoires à une catégorie de personnes selon leur privilèges sur le site ou leur login).

3.2.3. Configuration du fichier "webdefault.xml"

Le fichier comprend plusieurs sections.

La section suivante definit le filtre global pour tous les fichiers. Il vérifie la validité du site de demandé avant tout traitement par la servlet NVDCMS, dans le cas d'un site non valide une erreur est retournée au navigateur, dans le cas d'un site désactivé avec un visiteur n'aillant pas les droits nécessaires pour y entrer, une redirection est effectuée.

```
<filter>
<filter-name>GlobalFilter</filter-name>
<filter-class>org.nvdcms.cms.servlet.filter.GlobalFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>GlobalFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

La section "servlet" définit le servlet NVDCMS avec les différents fichiers de configuration (voir Section 2.3).

```
<servlet>
<servlet-name>CmsServlet</servlet-name>
<servlet-class>org.nvdcms.cms.servlet.http
.LegacyRequestProcessorWrapperServlet</servlet-class>
```

```
<init-param>
<param-name>logger-configuration</param-name>
<param-value>./etc/logKit.xml</param-value>
</init-param>
<init-param>
<param-name>component-role-configuration</param-name>
<param-value>./etc/roleConfig.xml</param-value>
</init-param>
<init-param>
<param-name>component-configuration</param-name>
<param-value>./etc/componentConfig.xml</param-value>
</init-param>

<load-on-startup>1</load-on-startup>
</servlet>
```

La partie :

```
<servlet-mapping>
<servlet-name>CmsServlet</servlet-name>
<url-pattern>*.xml</url-pattern>
</servlet-mapping>
```

Indique que pour l'extension *.xml le servlet "CmsServlet" prend en charge le traitement du fichier. D'autres extensions peuvent être ajoutées à l'aide de virgules par exemple :

```
<url-pattern>*.xml, *.XML</url-pattern>
```

pour les extensions xml et XML.

La partie :

```
<welcome-file-list>
<welcome-file>index.xml</welcome-file>
</welcome-file-list>
```

Indique le fichier par défaut, pris lorsqu'un répertoire est mentionné dans une URL. Plusieurs fichiers peuvent être spécifiés, le premier fichier qui existe sera renvoyé (ou éxécuté). Par exemple dans ce cas de figure, "http://site.com/" ira chercher "http://site.com/index.xml".

La partie :

```
<error-page>
<error-code>403</error-code>
<location>/index.xml</location>
</error-page>
```

Indique pour chaque code d'erreur la page web vers laquelle le navigateur est redirigé.

La partie :

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

Indique les règles de sécurité à appliquer pour les méthodes HTTP concernant les répertoires mentionnés en élément.

3.2.4. Configuration du fichier "admin.xml"

Le fichier de configuration est le suivant :

```

```

```
<Configure class="org.nvdcms.cms.jetty.CmsServer">
<Call name="addListener">
<Arg>
<New class="org.mortbay.http.SocketListener">
<Set name="Port">3406</Set>
<Set name="MinThreads">1</Set>
<Set name="MaxThreads">5</Set>
<Set name="MaxStopTimeMs">5000</Set>
<Set name="MaxIdleTimeMs">10000</Set>
</New>
</Arg>
</Call>

<Call name="addContext">
<Arg>/</Arg>
<Call name="addServlet">
<Arg>Admin</Arg>
<Arg>/*</Arg>
<Arg>org.nvdcms.cms.jetty.ManagementServlet</Arg>
<Put name="login">login</Put>
<Put name="pwd">pwd</Put>
</Call>
</Call>
</Configure>
```

La section "org.mortbay.http.SocketListener" indique la configuration de la socket d'écoute (ici sur le port 3046) , port normalement visible uniquement pour le réseau local.

La section suivante définit un unique servlet qui traite toutes les requêtes.

```
<Call name="addContext"> ...</Call>
```

Les deux paramètres, "login" et "pwd" sont utilisés pour l'authentification (en BASIC AUTHENTIFICATION HTTP) de la personne qui arrête le serveur HTTPD à l'aide de la requête à l'URL suivante :

```
http://x.x.x.x:3406/managementSTOP
```

3.2.5. Configuration webdav

Le module WebDAV correspond à un servlet permettant la gestion de fichiers à distance. Sa configuration est équivalente à la servlet NVDCMS. Il faut lui allouer une uri pour configurer le server http (pour l'exemple, on prendra l'uri /webdavURI)

Ajouter une context dans le serveur http, et définir le servlet comme élément qui traite toutes les requêtes.

```
<!--
WebDav Configuration
-->
<Call name="addContext">
<Arg>/webdavURI/*</Arg>
<Set name="ResourceBase">/www</Set>
<Call name="addServlet">
<Arg>WebDav</Arg>
<Arg>/</Arg>
<Arg>org.nvdcms.cms.Webdav</Arg>
<Put name="debug">0</Put>
<Put name="listings">true</Put>
<Put name="readonly">false</Put>
</Call>
</Call>
```

Toutes les requêtes commençant par /webdavURI seront traitées par le servlet WebDav, les paramètres d'initialisation sont :

- debug : définit le niveau de debug dans la console, à mettre à 0 pour la production
- listings : spécifie si le contenu des répertoires peut être listé
- readonly : spécifie si le servlet est en mode read only, seul la consultation des fichiers est autorisé, pas l'envoi.

L'accès se fait sur l'adresse suivante (le slash final est important) :

```
http://site.com/webdavURI/
```

3.2.6. Configuration iptables Linux 2.4

Sous Linux 2.4, la commande iptables permet de rediriger des requêtes d'un port vers un autre.

Les commandes à exécuter en tant que root sont :

```
# iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 8443
# iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080
# iptables -t nat -A OUTPUT -p tcp -s host1 --dport 80 -j REDIRECT -d host1
--to-port 8080
```

Les deux premières commandes redirigent le port 443 vers 8443 (HTTPS) et 80 vers 8080 (HTTP). La 3ème commande redirige le port 80 sur 8080 sur la machine elle-même (dont le nom est host1).

Puis pour vérifier que la commandes fonctionne il suffit de taper :

```
# iptables -t nat -L
```

3.3. Installation de certificat SSL

3.3.1. Création d'un certificat

Le JDK dispose en standard d'un outil pour la gestion des certificats : "keytool".

Les certificats sont stockés dans un fichier, chaque certificat est identifié par un alias.

Pour créer le certificat (avec par exemple l'alias 'demoServer'), et le fichier (de nom 'certificats.ks') qui va le contenir, exécuter la commande suivante :

```
keytool -genkey -keyalg RSA -alias demoServer -keystore certificats.ks
```

Le programme demande un mot de passe pour la gestion du fichier, puis diverses informations sur le propriétaire du certificat tel que le nom de l'Organisation, la localisation géographique, etc...

Attention : les informations "first and last name" correspondent au nom du site web (www.nvdcms.org par exemple).

Exemple d'exécution de la commande :

```
keytool -genkey -keyalg RSA -alias demoServer -keystore certificats.ks
Enter keystore password:  certifPWD
What is your first and last name?
[Unknown]:  www.nvdcms.org
What is the name of your organizational unit?
[Unknown]:  Computer Department
What is the name of your organization?
[Unknown]:  NVDCMS Org
What is the name of your City or Locality?
[Unknown]:  Paris
What is the name of your State or Province?
[Unknown]:  IDF
```

```
What is the two-letter country code for this unit?
[Unknown]:  FR
Is <CN=www.nvdcms.org, OU=Computer Department, O=Novadeck Org, L=Paris / IDF,
ST=IDF, C=FR> correct?
[no]:yes

Enter key password for <demoServer>
(RETURN if same as keystore password):  demoPWD
```

3.3.2. Faire signer son certificat

Il faut exporter le certificat dans un format plus standard que celui de JAVA à l'aide de la commande :

```
keytool -certreq -alias demoServer -keystore certificats.ks
```

Le programme demande le mot de passe du fichier de certificat plus celui du certificat et sort dans la console le certificat désiré.

Exemple d'execution de la commande :

```
keytool -certreq -alias demoServer -keystore certificats.ks
Enter keystore password:  certifPWD
Enter key password for <demoServer>:  demoPWD
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBvDCCASUCAQAwfDELMAkGA1UEBhMCRlIxDDAKBgNVBAgTA0lERjETMBEGA1UEBxMKU2FpbnQt
T3VlbjERMA8GA1UEChMITm92YWRlY2sxHDAaBgNVBAsTE0NvbXB1dGVyIERlcGFydG1lbnQxGTAX
BgNVBAMTEHd3dy5ub3ZhZGVjay5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALaxUXMI
RWQm1rlDPlXtEHHpjPtsXzCGhn251CjdktKf79rQMhCtAe1rvOaBUi5Hl5fKWF2eMxKeAYjkAF0Q
JWDLqOcG+xRcx7D8Cnteq7/MVB+CSW+9juy1nMButKIl93HZM3jrgXAI2NDHBSgZbbBkA3HwlMZI
lhm3+ba38ZgrAgMBAAGgADANBgkqhkiG9w0BAQQFAAOBgQBQ+zVUDYjsoqbeBWQ+DndwZX6qy/21
8g5sitQSVY3FCYvJ3YP+tjP5x9o8LS9xxVWZmhWPqQMPV8FPYvfrpDlO2seu51rWcBUuXOv+M1Iw
SJzT7Nk6y/gNjAM2ixK+eIGr0VBZ0pOcQ7qqbCz6K7NMPN6sTBn3+sBI/F69woPTpg==
-----END NEW CERTIFICATE REQUEST-----
```

Pour un certificat de test, l'organisme de certification Thawte propose gratuitement de signer les certificats par son serveur de test.

Attention : les certificats ainsi signés ne sont pas valable plus de 30 jours, et les navigateurs affichent un une fenêtre pour faire valider le certificat aux utilisateurs.

Aller à l'URL suivante : https://www.thawte.com/cgi/server/try.exe et sélectionner les valeurs par défaut des différents formulaires, jusqu'au formulaire contenant un textarea où il faudra copier-coller le certificat généré précedement :

```
-----BEGIN NEW CERTIFICATE REQUEST-----
...
-----END NEW CERTIFICATE REQUEST-----
```

Après validation, Thawte génére alors le certificat signé :

```
-----BEGIN CERTIFICATE-----
MIICoTCCAgqgAwIBAgIDBpuuMA0GCSqGSIb3DQEBBAUAMIGHMQswCQYDVQQGEwJa
QTEiMCAGA1UECBMZRk9SIFRFU1RJTkcgUFVSUE9TRVMgT05MWTEdMBsGA1UEChMU
VGhhd3RlIENlcnRpZmljYXRpb24xFzAVBgNVBAsTDlRFU1QgVEVTVCBURVNUMRww
GgYDVQQDExNUaGF3dGUgVGVzdCBDQSBSb290MB4XDTAyMTEyOTE3MTY0NFoXDTAy
MTIyMDE3MTY0NFowfDELMAkGA1UEBhMCRlIxDDAKBgNVBAgTA0lERjETMBEGA1UE
BxMKU2FpbnQtT3VlbjERMA8GA1UEChMITm92YWRlY2sxHDAaBgNVBAsTE0NvbXB1
dGVyIERlcGFydG1lbnQxGTAXBgNVBAMTEHd3dy5ub3ZhZGVjay5jb20wgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBALaxUXMIRWQm1rlDPlXtEHHpjPtsXzCGhn25
1CjdktKf79rQMhCtAe1rvOaBUi5Hl5fKWF2eMxKeAYjkAF0QJWDLqOcG+xRcx7D8
Cnteq7/MVB+CSW+9juy1nMButKIl93HZM3jrgXAI2NDHBSgZbbBkA3HwlMZIlhm3
+ba38ZgrAgMBAAGjJTAjMBMGA1UdJQQMMAoGCCsGAQUFBwMBMAwGA1UdEwEB/wQC
MAAwDQYJKoZIhvcNAQEEBQADgYEAcdjFMui2k9IpR94c67FIUkeOUAnyMIcmPN+f
```

```
ZlleUuZf4FL/d+6AQYPS/LuQ/QzwxTnrywvbJA51F1Kp1x1NiCwpkbK2uRX/86LN
g2XsfQT8jOnuZKCdjwXz7PMWRfs9yKkCymdyzx0Glk9y8kv/PNHL3qZHfMxzF/Hq
3h0F1JU=
-----END CERTIFICATE-----
```

Il sufit de recopier le certificat signé dans un fichier "thawte.test". Attention : le fichier doit se terminer par une ligne vide.

3.3.3. Importer le certificat signé

Le certificat maintenant signé par un organisme de certification, il est nécessaire de l'importer dans le fichier de certificats qui sera utilisé par le serveur.

Le programme 'keytool' utilisé avec l'option '-import' demande le mot de passe du fichier de certificats, et demande de valider le certificat de Thawte. Utiliser la commande suivante :

```
keytool -import -keystore certificats.ks -file thawte.test
Enter keystore password:  certifPWD
Owner: CN=www.nvdcms.org, OU=Computer Department, O=NVDCMS Org, L=Saint-Ouen,
ST=IDF, C=FR
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 69bae
Valid from: Fri Nov 29 18:16:44 CET 2002 until: Fri Dec 20 18:16:44 CET 2002
Certificate fingerprints:
MD5:  FA:77:E1:E4:DD:ED:6A:CC:85:9D:56:09:AA:91:4F:1B
SHA1: EA:E2:54:6D:9D:A2:35:19:47:F1:B9:A9:FE:E8:D1:10:C0:90:8B:39
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

3.3.4. Configurer httpd

Jetty a besoin de la librairie JSSE (téléchargeable à l'adresse. http://java.sun.com/products/jsse/ pour les anciens JDK , mais livrée en standard sur les JDK 1.4.X) qu'il faut ajouter au classpath.

De plus il faut modifier le fichier de configuration Jetty pour lui préciser le fichier de certificat à utiliser et les mots de passe associés.

```
<Call name="addListener">
<Arg>
<New class="org.mortbay.http.SunJsseListener">
<Set name="Port">8443</Set>
<Set name="MinThreads">5</Set>
<Set name="MaxThreads">100</Set>
<Set name="MaxIdleTimeMs">30000</Set>
<Set name="LowResourcePersistTimeMs">2000</Set>
<Set name="Keystore"><SystemProperty
name="jetty.home" default="."/>/etc/certificats.ks</Set>
<Set name="Password">certifPWD</Set>
<Set name="KeyPassword">demoPWD</Set>
</New>
</Arg>
</Call>
```

3.4. Démarrage/Arrêt

L'exécution du script de démarrage doit être effectuée avec l'utilisateur 'wwwrun'

Le démarrage : dans le répertoire /usr/local/jetty/bin

```
./nvdcms.sh start
```

L'arrêt: dans le répertoire /usr/local/jetty/bin

```
./nvdcms.sh stop
```

3.5. Mise en place des logs

3.5.1. Configuration de syslog sur RED HAT 7.0+

3.5.1.1. Editer /etc/rc.d/init.d/syslog

Modifier :

```
case "$1" in
start)
echo -n "Starting system logger: "
# we don't want the MARK ticks
daemon syslogd -m 0
RETVAL=$?
```

En :

```
case "$1" in
start)
echo -n "Starting system logger: "
# we don't want the MARK ticks
daemon syslogd -r -m 0
RETVAL=$?
```

Le "syslogd -r" indique une réception de messages envoyés depuis le réseau.

3.5.1.2. Editer le fichier /etc/syslog.conf

Choisir un élément de { local0, local1, ..., local7} n'étant pas déjà pris (généralement "local0" est disponible) et ajouter la ligne :

```
local0.*        /var/log/nvdcms/http.log
```

(ou alors tout autre emplacement pour stocker les fichiers de logs)

Penser à supprimer le fait que les logs vont aussi dans /var/log/messages avec la ligne :

```
*.info;mail.none;authpriv.none;local0.none      /var/log/messages
```

3.5.1.3. Redémarrer syslog

Avec le compte 'root' :

```
/etc/rc.d/init.d/syslog restart
```

3.5.1.4. Configurer la rotation de logs pour éviter d'avoir des logs trop volumineux

dans /etc/logrotate.d/syslog ajouter :

```
# logs from cluster HTTP
/var/log/nvdcms/http.log {
postrotate
/usr/bin/killall -HUP syslogd
endscript
}
```

3.5.2. Configuration de syslog sur Suse 7.2 et 8.x

3.5.2.1. Autoriser la réception de messages envoyés depuis le réseau

Editer le fichier /etc/rc.config sur Suse 7.2 (sur Suse 8.x le fichier à modifier est /etc/sysconfig/syslog), et rechercher la ligne contenant SYSLOGD_PARAMS (qui définit les paramètres du daemon syslog), ajouter l'option '-r' si elle n'y est pas (par défaut, cette variable est vide).

```
SYSLOGD_PARAMS=" -r "
```

3.5.2.2. Editer le fichier /etc/syslog.conf

Les logs sont envoyés au daemon syslog avec comme 'facility' Local0. 'Facility' est une sorte de flag dans l'envoi des données au daemon, permettant à ce dernier de répartir plus efficacement les différents messages qu'il reçoit. Pour envoyer tous les messages reçus sur Local0 vers un emplacement qui lui est spécifique, il faut ajouter la ligne :

```
local0.*        /var/log/nvdcms/http.log
```

(ou alors tout autre emplacement pour stocker les fichiers de logs)

Penser à supprimer le fait que les logs vont aussi dans /var/log/messages avec la ligne :

```
*.info;mail.none;authpriv.none;local0.none     /var/log/messages
```

Afin que les messages de logs http n'aillent pas pollués les autres fichiers de logs du systèmes.

3.5.2.3. Redémarrer syslog

Avec le compte 'root' :

```
/etc/rc.d/init.d/syslog restart
```

3.5.2.4. Configurer la rotation de logs pour éviter d'avoir des logs trop volumineux

On peut faire en sorte que passer une certaine taille de fichier, le fichier de logs soit remplacé par un nouveau fichier, et sauvegardé sous un autre nom. Editer le fichier /etc/logfiles et ajouter à la fin :

```
# HTTP LOGS FROM Jetty
/var/log/nvdcms/http.log   +10240k  644     wwwrun.root     syslog
```

La ligne suivante indique que le fichier /var/log/nvdcms/http.log doit être remplacé s'il a atteint la taille de 10 Mo, qu'il se voit affecté les droits 644 (à la base, il appartient au user root), qu'il passe sous la propriété du user wwwun et du groupe root, et qu'il dépend du service syslog.

3.5.3. Configuration de "logKit.xml" pour envoyer les logs vers syslog

Editer le fichier logKit.xml dans nvdcms/etc definir les logs http sur le nouveau canal ouvert.

Editer la partie factories (En début de fichier). Définir une "factory" pour l'envoi de données sur syslog

```
<factory class="org.nvdcms.cms.avalon.logger.factory.DatagramTargetFactory"
type="datagram"/>
```

Editer la partie targets. Définir une destination pour l'envoi de données sur syslog.

```
<datagram id="syslog">
<address hostname="10.0.1.92" port="514" />
```

```
<format type="syslog" facility="128"/>
</datagram>
```

L'attribut "id" associe un nom (qui est unique) à cette destination. L'attribut "hostname" correspond à la machine qui recevra les données (elle doit dispose d'un daemon syslog en écoute sur le réseau. L'attribut "port" correspond au port de syslog (invariable). L'attribut "facility" correspond au niveau de log "LOCAL0" de syslog (invariable).

Définir cette destination pour les logs HTTP, Editer la partie "categories", et ajouter

```
<category name="tools.LogLine" log-level="INFO">
<log-target id-ref="syslog"/>
</category>
```

Chapitre 4. EJB

4.1. Weblogic 6.1

4.1.1. Installation

Par défaut l'outil d'outil d'installation fonctionne avec interface graphique. Sous Linux et Solaris, l'installation peut se faire en mode console avec les commandes:

- Linux : /usr/local/jdk/bin/java -cp weblogic610sp2_generic.zip install -i console
- Solaris : sh weblogic610sol.bin -i console

Les renseignements à fournir lors de l'installation sont simples : répertoire d'installation, mot de passe administrateur, installation complète (avec exemples) ou minimale, et divers autres informations à laisser aux valeurs par défaut.

L'installation doit être effectuée avec le compte administrateur (admin).

4.1.2. Arborescence

Le serveur Weblogic peut être installé dans le répertoire (dans ce cas pensé à créer le répertoire et lui donner les bons droits pour que le user admin puisse y écrire) :

```
/usr/local/bea/
```

Ce dernier comprend les éléments suivants :

- jdk131 : répertoire contenant le JDK 1.3.1 (sauf sous Linux) pour l'exécution du serveur
- wlserver6.1 : répertoire principal de l'application
- license.bea : fichier XML de licence

En mode "cluster", Weblogic 6.1 fonctionne par domaine. Lors de l'installation, il faut préciser un nom de domaine (par défaut "mydomain") et un nom de serveur (par défaut "myserver") pour générer un fichier de configuration initial.

Le répertoire

```
/usr/local/bea/wlserver6.1/config/
```

contient l'ensemble des répertoires de travail des différents domaines.

Pour le domaine par défaut, le répertoire de travail est

```
/usr/local/bea/wlserver6.1/config/mydomain/
```

et contient :

- SerializedSystemIni.dat : fichier binaire indispensable pour la gestion des mots de passe WebLogic
- applications/ : répertoire de déploiement des applications gérées par weblogic
- config.xml : fichier de configuration du serveur
- config.xml.FROM_INSTALLER : copie du fichier de configuration installé par défaut
- config.xml.booted : copie du fichier de configuration du dernier démarrage
- fileRealm.properties : fichier contenant les logins des utilisateurs propre à WebLogic

- logs/ : répertoire des fichiers de logs
- startWebLogic.sh et startWebLogic.cmd : script de démarrage de WebLogic respectivement shell unix et commande windows

Les fichiers SerializedSystemIni.dat et fileRealm.properties sont essentiels, car ils contiennent les mots de passe cryptés (il est préférable d'en garder une copie en sureté).

Chaque serveur weblogic a un nom "unique dans le cluster" (myserver étant la valeur par défaut).

4.1.3. Configuration

Le fichier de configuration "config.xml" comprend plusieurs sections à éditer. Une documentation online est disponible : http://edocs.bea.com/wls/docs61/config_xml/index.html.

Remarques :

- pour l'ensemble des élements, l'attibut "Name" est obligatoire, il représente un identifiant unique permettant de localiser chaque ressource (WebLogic fait un usage intensif des Mbeans)
- l'attribut "Targets" représente, dans chaque cas, la liste des serveurs où sera déployée la ressource (liste de noms séparés par une virgule)

4.1.3.1. Définition du serveur (bloc requis)

Editer la section suivante :

```
<Server ListenPort="7001"
Name="myserver"
NativeIOEnabled="true"
HttpdEnabled="false"
StdoutSeverityLevel="64" />
```

- HttpdEnabled : lance le serveur web (option non prise en compte en mode simple serveur)
- StdoutSeverityLevel : niveau de sortie debug à la console
- NativeIOEnabled : active/désactive le performance pack

4.1.3.2. Définition d'un pool de connexions à une base de données

Exemple de définition d'un pool de connexion à une base de données postgresql :

```
<JDBCConnectionPool CapacityIncrement="2"
DriverName="org.postgresql.Driver" InitialCapacity="3"
LoginDelaySeconds="5" MaxCapacity="10" Name="jdbcConnection"
Properties="user=mylogin;password=mypassword"
RefreshMinutes="1" ShrinkPeriodMinutes="15"
ShrinkingEnabled="true" SqlStmtProfilingEnabled="false"
Targets="myserver" TestConnectionsOnReserve="false"
TestTableName="sites" URL="jdbc:postgresql://127.0.0.1:5432/nvd"/>
```

Le nom attribué à la connection avec l'attribut Name="jdbcConnection" est utilisé par la suite dans la définition d'un datasource. Le pool de connexion est définie à l'aide des attributs "InitialCapacity" "MaxCapacity".

- SqlStmtProfilingEnabled : permet d'enregistrer sur disque dans un fichier XML la liste des requêtes SQL envoyées à la base

4.1.3.3. Définition de datasource

Le datasource suivant utilise le pool de connexion "jdbcConnection" défini ci-dessus.

```
<JDBCDataSource JNDIName="datasourceJdbcPool1"
Name="datasource-Pool1"
PoolName=" jdbcConnection" Targets="myserver"/>
```

- JNDIName : nom JNDI du datasource , jdbcPool1 pour la plateforme Novadeck CMS
- PoolName : nom du pool de connexion à utiliser

4.1.3.4. Définition des EJB à deployer

```
<Application Deployed="true" Name="entities" Path="/path/to/ejb">
<EJBComponent Name="entitiesJAR" Targets="carbone" URI="entities.jar"/>
</Application>
```

- Path: répertoire contenant les fichiers
- URI : nom du fichier à déployer

4.1.3.5. Définition des Topics JMS

Pour un bon fonctionnement du CMS, il est nécessaire de declarer les "topics" JMS suivants (les noms "topic/DISTRIBUTED_MAP" et "queue/NVDCRON" doivent correspondre aux déclaration JMS effectuées dans le fichiers de configuration du CMS).

```
<JMSServer Name="CMSJMS" Targets="myserver">
<JMSTopic JNDIName="topic/DISTRIBUTED_MAP" Name="distributedmap"/>
<JMSTopic JNDIName="queue/NVDCRON" Name="nvdcron"/>
</JMSServer>
```

4.1.4. Démarrage/Arrêt

Par défaut le fichier de lancement de l'application demande le mot de passe administrateur lors de l'execution du script. Pour éviter d'avoir à saisir le mot de passe, il faut mettre à jour les scripts de démarrage, en modifiant la ligne :

```
WLS_PW=
```

par

```
set WLS_PW=LeMotDePasseAdministrateur
```

La ligne

```
set STARTMODE=true
```

indique que le serveur est lancé en mode de production. En mode non production, le serveur déploit automatiquement les applications contenues dans le répertoire "application/".

Les variables d'environnement fournies à la JVM au lancement de l'application sont :

- weblogic.Domain : nom du domaine dont fait partie le serveur que l'on veut lancer
- weblogic.Name : nom du serveur à lancer
- weblogic.management.password : mot de passe administrateur (voir WLS_PW)
- weblogic.ProductionModeEnabled : mode production actif/désactif (voir STARTMODE)

En mode "simple serveur", une interface web d'administration est accessible à l'URL suivante :

```
http://serveur:7001/console
```

(login / mot de passe requis).

4.2. JBoss 3.0.x

4.2.1. Installation

JBoss contient 3 configurations (dans le répertoire $JBOSS_HOME/server/) : 'all', 'minimal' et 'default'.

Pour installer la configuration nécessaire pour le fonctionnement du logciel NVD-CMS, la configuration 'all' - qui contient toutes les services JBoss - est utilisé. Pour ce faire, copier les répertoires

```
cp -a $JBOSS_HOME/server/all $JBOSS_HOME/server/nvdcms
```

JBoss construit son classpath automatiquement, tout fichier dans "$JBOSS_HOME/server/nvdcms/lib/" en fera partie. JBoss essaie de déployer tous les fichiers/dossiers contenus dans le répertoire "$JBOSS_HOME/server/nvdcms/deploy", cela concerne aussi bien les fichiers de configuration que les librairies exécutables.

L'installation doit être effectuée avec le compte administrateur (admin).

4.2.2. Configuration

4.2.2.1. Les fichiers de configuration

Les fichiers de configuration du serveur se trouvent dans :

```
$JBOSS_HOME/server/nvdcms/conf
```

Les fichiers suivants sont inutiles pour la plate forme NVD-CMS, et peuvent être supprimés :

```
auth.conf
axis-config.xml
jboss-minimal.xml
```

4.2.2.2. Les services

L'ensemble des services lancés par JBoss se trouvent dans le répertoire

```
$JBOSS_HOME/server/nvdcms/deploy
```

Les services suivants peuvent être supprimés :

```
cluster-service.xml
counter-service.xml
ejb-management.jar
hsqldb-service.xml
iiop-service.xml
jbossha-httpsession.sar
jbossmq-destinations-service.xml
jboss-net.sar
jbossweb-ejb.jar
jmx-ejb-adaptor.jar
jmx-ejb-connector-server.sar
mail-service.xml
properties-service.xml
scheduler-service.xml
farm-service.xml ( depuis la version 3.0.6)
schedule-manager-service.xml ( depuis la version 3.0.6)
```

La première chose à configurer est la connexion base de données. Plusieurs fichiers de configuration sont fournis avec l'installation de JBoss (dans "$JBOSS_HOME/docs/examples/jca/"). Il suffit de prendre celui qui correspond au serveur de base de données et le placer dans le répertoire "$JBOSS_HOME/server/nvdcms/deploy".

Cherchez le l'élement XML dans le fichier sélectionné :

```
<mbean code="org.jboss.resource.connectionmanager.RARDeployment" ... >
```

Indiquez les bonnes valeurs de connection ('url', 'driver class', 'user', 'password') à la base de données en donnant comme nom jndi : "jdbcPool1".

Par exemple pour une connexion à une base de données Postgresql :

```
<mbean code="org.jboss.resource.connectionmanager.RARDeployment"
name="jboss.jca:service=LocalTxDS,name=PostgresDS">
<attribute name="JndiName">jdbcPool1</attribute>
<attribute name="ManagedConnectionFactoryProperties">
<properties>
<config-property name="ConnectionURL" type="java.lang.String"
>jdbc:postgresql://10.0.0.7:5432/nvdcms</config-property>
<config-property name="DriverClass" type="java.lang.String"
>org.postgresql.Driver</config-property>
<config-property name="UserName" type="java.lang.String"
>dbuser</config-property>
<config-property name="Password" type="java.lang.String"
>dbpasswd</config-property>
</properties>
</attribute>
</mbean>
```

Ajuster le classpath pour que les EJB soient correctement deployés, il faut ajouter dans le répertoire "$JBOSS_HOME/server/nvdcms/lib" les librairies suivantes (en plus de la librairie contenant le driver JDBC de la connexion base) :

```
avalon-scratchpad.jar
avalon-framework.jar
nvdcms.jar
nvdtools.jar
logKit.jar
```

Deployer les EJB : copier les fichiers dans le répertoire "$JBOSS_HOME/server/nvdcms/deploy" (JBoss reconnait qu'il s'agit d' EJB et les deploit automatiquement) :

```
bmp.jar
cmp.jar
sessions.jar
```

4.2.2.3. Configuration du serveur JMS

Le dossier "$JBOSS_HOME/server/nvdcms/deploy" contient un fichier nommé "user-service.xml" (normalement dépourvu d'informations). Ce fichier est nécessaire pour configurer le serveur JMS.

Le logiciel NVD-CMS nécessite la configuration pour un "Topic" et une "Queue" suivante (à placer dans le fichier) :

```
<mbean code="org.jboss.mq.server.jmx.Topic"
name="jboss.mq.destination:service=Topic,name=DISTRIBUTED_MAP">
<depends optional-attribute-name="DestinationManager"
>jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue"
name="jboss.mq.destination:service=Queue,name=NVDCRON">
<depends optional-attribute-name="DestinationManager"
>jboss.mq:service=DestinationManager</depends>
</mbean>
```

Le logiciel NVD-CMS utilise des connexions au serveur pour la gestion de topic/queue JMS. JBoss fournit 4 implémentations de ces connexions (dans "$JBOSS_HOME/nvdcms/deploy/jbossmq-service.xml", voir la partie "Invocation Layers").

Il suffit de choisir l'une d'entre elles, et d'aliaser leur nom JNDI sur les noms utilisés par le logiciel NVD-CMS on peut commenter celles qui ne seront pas utilisées.

Par exemple pour prendre l'implémentation "OIL" :

```
<mbean code="org.jboss.naming.NamingAlias"
name="jboss.mq:service=NamingAlias,fromName=javax.jms.QueueConnectionFactory">
<attribute name="ToName">XAConnectionFactory</attribute>
<attribute name="FromName">javax.jms.QueueConnectionFactory</attribute>
</mbean>
<mbean code="org.jboss.naming.NamingAlias"
name="jboss.mq:service=NamingAlias,fromName=javax.jms.TopicConnectionFactory">
<attribute name="ToName">XAConnectionFactory</attribute>
<attribute name="FromName">javax.jms.TopicConnectionFactory</attribute>
</mbean>
```

4.2.2.4. Optimisations

Il est possible de paramétrer le cache et le fonctionnement du serveur entre chaque transaction en éditant le le fichier

```
$JBOSS_HOME/nvdcms/conf/standardjboss.xml
```

Ce fichier comporte des paramètres pour chaque type de EJB (CMP 1.x et 2.x, BMP, Session) (voir http://www.jboss.org/online-manual/HTML/ch07s16.html).

Supprimer les éléments :

```
<interceptor>org.jboss.ejb.plugins.LogInterceptor</interceptor>
<interceptor>org.jboss.ejb.plugins.SecurityInterceptor</interceptor>
```

"LogInterceptor" est inutile en production, il permet de logger chaque appel de méthode sur les EJB. "SecurityInterceptor" vérifie les permissions lors d'appel EJB au sens de la norme, Cette fonctionnalité est inexploitable avec le logiciel NVD-CMS.

Ajuster les valeurs des caches selon les besoins, par exemple le sous arbre :

```
<container-cache-conf>
<cache-policy
>org.jboss.ejb.plugins.LRUEnterpriseContextCachePolicy</cache-policy>
<cache-policy-conf>
<min-capacity>50</min-capacity>
<max-capacity>1000000</max-capacity>
<overager-period>300</overager-period>
<max-bean-age>600</max-bean-age>
<resizer-period>400</resizer-period>
<max-cache-miss-period>60</max-cache-miss-period>
<min-cache-miss-period>1</min-cache-miss-period>
<cache-load-factor>0.75</cache-load-factor>
</cache-policy-conf>
</container-cache-conf>
```

Dans le cas ou le JBoss est seul à modifier les données de la base de données, on peut configurer le serveur pour qu'il ne re-charge pas les données à chaque transaction en précisant :

```
<commit-option>A</commit-option>
```

au lieu de la valeur par défaut qui doit être :

```
<commit-option>B</commit-option>
```

Chapitre 5. Base de données

5.1. Oracle

5.1.1. Installation

Lors de l'installation Oracle il est recommandé de créer une base de données dédiée à l'application avec comme nom 'NVDCMS' par exemple. Il est vivement recommandé d'utiliser l'encoding UTF8 pour le stockage des caractéres.

Exemple de script PSQL de création de la base de données pour Oracle 9i :

```
connect SYS/change_on_install as SYSDBA
set echo on
spool /u01/app/oracle/product/9.0.1/assistants/dbca/logs/CreateDB.log
startup nomount pfile="/u01/app/oracle/admin/nvdora/scripts/init.ora";

CREATE DATABASE nvdora

MAXINSTANCES 1
MAXLOGHISTORY 1
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXDATAFILES 100

DATAFILE '/u01/app/oracle/oradata/nvdora/system01.dbf'
SIZE 325M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED

UNDO TABLESPACE "UNDOTBS"
DATAFILE '/u01/app/oracle/oradata/nvdora/undotbs01.dbf'
SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED

CHARACTER SET UTF8
NATIONAL CHARACTER SET AL16UTF16

LOGFILE GROUP 1 ('/u01/app/oracle/oradata/nvdora/redo01.log') SIZE 100M,
GROUP 2 ('/u01/app/oracle/oradata/nvdora/redo02.log') SIZE 100M,
GROUP 3 ('/u01/app/oracle/oradata/nvdora/redo03.log') SIZE 100M;
spool off
exit;
```

Dans le script ci dessus, nous créons un tablespace 'UNDOTBS' pour gérer les 'undos' et trois tablespaces pour les 'redos'.

Il est préférable d'installer Oracle en utilisant les arborescences standards (/u01/app, /u01/app/oracle/oradata) recommandées par Oracle.

Changer les mots de passe des comptes administrateurs existants :

```
ALTER USER system IDENTIFIED by password;
```

```
ALTER USER sys IDENTIFIED by password;
```

Il est préférable de créer un tablespace (NVD) dédié aux données de la plate forme :

```
CREATE TABLESPACE nvd
DATAFILE '/u01/app/oracle/oradata/nvdora/nvd.dbf' SIZE 50M
DEFAULT STORAGE (
INITIAL 50K
NEXT 50K
MINEXTENTS 2
MAXEXTENTS UNLIMITED
```

```
PCTINCREASE 0);
```

(où nvdora est le nom de sous-domaine donnée à la base) Ici le tablespace dispose d'une taille initiale de 50 Mo et se trouve stocké dans le path '/u01/app/oracle/oradata/nvdora/nvd.dbf'.

Créer un compte utilisateur (NVD) pour la gestion des données :

```
CREATE USER nvd IDENTIFIED by nvdpassword;
GRANT dba TO nvd;
ALTER USER nvd DEFAULT TABLESPACE 'nvd';
```

Ici nous créons l'utilisateur NVD avec le mot de passe NVDPASSWORD et nous lui attribuons des droits connect, create session et DBA *(pour plus de raffinement dans les droits attribués à l'utilisateur NVD, se reporter à la documentation Oracle)*. Par ailleurs, nous affectons par défaut le 'tablespace' nvd à l'utilisateur.

Supprimer les comptes et tables non utilisés et installés par défaut par oracle. Créer la structure des tables à partir du fichier de dump fourni en utilisant le compte utilisateur 'NVD'.

5.1.2. Arborescence

Le répertoire :

```
/u01/app/oracle/oradata/nvdora/
```

comprend les fichiers de la base de données NVDCMS (notamment le tablespace 'nvd.dbf').

Les binaires oracle sont situés dans les répertoire (pour la version 9.0.1) :

```
/u01/app/oracle/product/9.0.1/bin
```

5.1.3. Configuration

Le fichier :

```
/etc/oratab
```

contient la liste des bases de données gérées par Oracle.

Pour la base de données NVDCMS la ligne suivante doit figurer dans le fichier pour mentionner la prise en compte de la base NVDCMS :

```
NVDCMS:/u01/app/oracle/product/9.0.1:Y
```

5.1.4. Démarrage/Arrêt

Le script de démarrage/arrêt est le suivant (à exécuter en tant que 'root')

```
export ORACLE_HOME=/u01/app/oracle/product/9.0.1/
case "$1" in
start) echo "Starting oracle databases..."
sleep 2
su - oracle -c "$ORACLE_HOME/bin/dbstart"
su - oracle -c "$ORACLE_HOME/bin/lsnrctl start"
;;

stop) echo "Stoping oracle databases..."
sleep 2
su - oracle -c "$ORACLE_HOME/bin/lsnrctl stop"
su - oracle -c "$ORACLE_HOME/bin/dbshut"
;;
```

- $ORACLE_HOME/bin/dbstart : est un script oracle qui se charge de démmarer les bases de données mentionnées dans le fichier /etc/oratab

- $ORACLE_HOME/bin/dbshut : est un script oracle qui se charge d'arreter les bases de données mentionnées dans le fichier /etc/oratab
- $ORACLE_HOME/bin/lsnrctl start | stop : est un script oracle qui permet la mise en route et l'arrêt du 'Listener' Oracle qui se charge de prise en compte des connexions IP à la base via le réseau.

5.1.5. Tips

Pour afficher tous les index du tablespace NVD :

```
SELECT * FROM all_indexes WHERE TABLESPACE_NAME='NVD';
```

sqlplus (mode Console) :

- SHOW ALL : affiche les variables d'environnements de la console
- SELECT * FROM tabs : affiche les tables appartenant au user courant
- DESC nom_de_table_ou_de_vue : donne la définition de l'élément

Pour pouvoir visionner l'état de la base (avec Oracle 9i et un UNDO_TABLESPACE) à un instant donnée (ne fonctionne pas dans pour tous les cas de figure, comme le cas d'un DROP de table).

```
Dbms_Flashback.Enable_At_Time('15-OCT-02 11:59:00');
```

Pour restaurer une table 'mytable' après un DELETE et COMMIT :

```
DECLARE
CURSOR c_mytable IS
SELECT * FROM mytable;
v_row c_mytable%ROWTYPE;
BEGIN
Dbms_Flashback.Enable_At_Time(SYSDATE - 10/1440);
OPEN c_mytable;
Dbms_Flashback.Disable;

LOOP
FETCH c_mytable INTO v_row;
EXIT WHEN c_mytable%NOTFOUND;
INSERT INTO mytable VALUES
(v_row.ID, v_row.OBJECTSINSTANCES_ID,
v_row.CONTENT, v_row.REF_ID,
v_row.PLACE, v_row.LANGUAGE);
END LOOP;
CLOSE c_mytable;
COMMIT;
END;
/
```

5.2. Postgresql

5.2.1. Installation

Créer un utilisateur Unix 'postgres' pour l'exécution de la base de données. Télécharger les sources Suivre la procédure d'installation

```
./configure --enable-multibyte --with-java --prefix=/usr/local/pgsql
make
make install
```

options de configure :

- --enable-multibyte : (depuis la version 7.1) - support UNICODE
- --with-java : compile le driver JDBC. La compilation nécessite la variable d'environnement ANT qui pointe vers un fichier ant.sh
- --prefix=/usr/local/pgsql : indique que l'installation se fera dans le répertoire /usr/local/pgsql

5.2.2. Arborescence

Les fichiers postgresql sont généralement installés dans le répertoire :

```
/usr/local/pgsql
```

Cet emplacement peut être modifié lors de l'étape de construction du Makefile à l'aide de la commande 'configure' mentionnée précédement.

5.2.3. Configuration

5.2.3.1. Création du système

```
path_to_postgresql/bin/initdb -D chemin_de_stockage_du_systeme/ -E UNICODE
```

construit le système avec comme encodage par défaut : UTF-8 (UNICODE)

5.2.3.2. Paramétrer le daemon postgres

dans le fichier chemin_de_stockage_du_systeme/postgresql.conf ajouter :

```
fsync = on
tcpip_socket = true
sql_inheritance = false
silent_mode = true
stats_start_collector = false
```

- fsync : écriture systématique sur le disque (en mode exploit mettre cette valeur à 'off')
- tcpip_socket : autorise les connections via TCP/IP
- sql_inheritance : refuse l'héritage de table (optionnel)
- silent_mode : mode daemon (pas de debug dans la console)
- stats_start_collector : désactive le processus de statistiques sur les requêtes faites en base.

5.2.3.3. Création de la base de données NVDCMS

Le démon postgresql étant lancé, exécuter la commande :

```
path_to_postgresql/bin/createdb -E UNICODE nvdcms
```

Elle crée une nouvelle base dans le système courant en utilisant l'encodage UTF-8.

5.2.4. Démarrage/Arrêt

5.2.4.1. Démarrage

Se connecter avec le compte Unix 'postgres', exécuter la commande :

```
path_to_postgresql/bin/postmaster -D chemin_de_stockage_du_systeme/
```

5.2.4.2. Arrêt

Avec le compte Unix 'postgres'

```
path_to_postgresql/bin/pg_ctl stop -D chemin_de_stockage_du_systeme/
```

5.2.5. Dump/Restore

5.2.5.1. Dump

Se connecter avec le compte Unix 'postgres', exécuter la commande :

```
path_to_postgresql/bin/pg_dump -h MACHINE -f dump.sql.gz -Z 3 NOM_DE_BASE
```

crée un fichier gzippé avec les scripts de créations des différents éléments du système (tables/index/vues/..) et les données dans un format propriétaires, pour un export en format sql ajouter l'option -d.

5.2.5.2. Restore

Avec le compte Unix 'postgres'

```
path_to_postgresql/bin/psql -h MACHINE -f dump.sql NOM_DE_BASE
```

5.2.6. Tips

Pour défragmenter la base :

```
vaccum / vaccum analyze
```

psql (mode console) :

```
\l liste les différentes databases du sytème
\dt affiche la liste des tables dans la base
\dv affiche la liste des vues dans la base
\d nom_de_table_ou_de_vue donne la définition de l'élément
```

Pour réindexer une table:

```
REINDEX TABLE nom_de_la_table;
```

utile pour les tables ayant beaucoup d'écriture/suppression.

Chapitre 6. DNS

6.1. BIND 8

6.1.1. Arborescence

Les fichiers de configuration serveur de bind8 sont placés dans les emplacements suivants :

- /etc/named.conf : indique les répertoires de travail (généralement /var/named)
- /var/named/ : comprend les fichiers de configuration de chaque domaine

6.1.2. Configuration

6.1.2.1. named.conf

Le fichier se décompose en 3 parties :

- la section options{} qui contient les options générales du serveur de noms
- la section qui pour une classe d'adresse IP associe un nom de domaine unique.

```
zone "xxx.xxx.xxx.in-addr.arpa" {
type master;
file "/var/named/db.xxx.xxx.xxx";
};
```

- la section zone qui associe un nom de domaine à son fichier de configration :

```
zone "nom_de_domaine" {
type master;
file "/var/named/db.nom_de_domaine";
};
```

Chaque zone renvoie vers son fichier de configuration situé dans le répertoire /var/named (par exemple ici /var/named/db.xxx.xxx.xxx ou /var/named/db.nom_de_domaine).

Exemple de fichier named.conf :

```
options {

# The directory statement defines the name server's
# working directory

directory "/var/named";
# The cleaning-interval statement defines the time interval
# in minutes for periodic cleaning. Default is 60 minutes.
# By default, all actions are logged to /var/log/messages.

cleaning-interval 120;

# Name server statistics will be logged to /var/log/messages
# every <statistics-interval> minutes. Default is 60 minutes.
# A value of 0 disables this feature.

statistics-interval 0;

# If notify is set to yes (default), notify messages are
```

```
# sent to other name servers when the the zone data is
# changed. Instead of setting a global 'notify' statement
# in the 'options' section, a separate 'notify' can be
# added to each zone definition.
auth-nxdomain yes;
notify no;
};

# The following three zone definitions don't need any modification.
# The first one defines localhost while the second defines the
# reverse lookup for localhost. The last zone "." is the
# definition of the root name servers.

zone "localhost" in {
type master;
file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
type master;
file "127.0.0.zone";
};

zone "." in {
type hint;
file "root.hint";
};
zone "everydeck.com" {
type master;
file "/var/named/db.everydeck.com";
};
```

6.1.2.2. /var/named/*

Chaque domaine dispose de son fichier de configuration.

```
$TTL    4d
@       IN      SOA     dns1.everydeck.com.    admin.everydeck.com. (
200212311700;Serial
1d;Refresh
2h;Retry
4w;Expire
4m;Minimum
)


@                   IN      NS          dns1.everydeck.com.
@                   IN      NS          dns2.everydeck.com.
@                   IN      MX      10  mail1.everydeck.com.
@                   IN      MX      20  mail2.everydeck.com.
everydeck.com.      IN      NS          dns1.everydeck.com.
everydeck.com.      IN      NS          dns2.everydeck.com.
dns2.everydeck.com.          A          195.167.192.166
dns1.everydeck.com.          A          195.167.192.165
everydeck.com.               A          195.167.192.165
*.everydeck.com.             A          195.167.192.165
pop.everydeck.com.           A          195.167.192.167
```

TTL : indique le Time To Leave (temps d'expiration des caches des serveurs DNS).

La ligne contenant les infos MX indique les serveur de mail qui gèrent le domaine indiqué ici.

La ligne contenant *.everydeck.com indique que tous les sous-domaine s d'everydeck.com auront pour IP l'adresse 195.167.192.165.

6.1.3. Démarrage/Arrêt

Pour démarrer le serveur de nom, taper la commande suivante à l'aide du compte utilisateur "root" :

```
#/etc/init.d/named start
```

Pour arrêter le serveur de nom, taper la commande suivante à l'aide du compte utilisateur "root" :

```
#/etc/init.d/named stop
```

Chapitre 7. Analyse de Logs HTTP avec JXLA

7.1. Stockage des requêtes HTTP

Logs HTTP sont centralisés. Tous les serveurs web de la la plate-forme envoient leurs logs HTTP sur un démon SYSLOG -disponible sur tout serveur unix-. Chaque serveur web utilise le module SysLog du 'log toolkit apache'.

Ce dernier est en charge de stocker les requêtes sur disque, d'ajouter la date et l'heure de la reqête (impossible à modifier ce comportement), d'effectuer la rotation des fichiers de logs lorsqu'ils deviennent trop volumineux et leur historique.

7.2. JXLA

Le programme qui analyse les logs est JXLA (http://jxla.novadeck.org/), auquel a été ajouté un module spécique à la plate-forme pour la reconnaissance des utilisateurs, et la répartition des logs par nom de sites.

Le programme peut être installé dans n'importe quel répertoire, pour la suite on prendra /home/logs/ comme référence.

Le programme génère des fichiers XML dans un répertoire du site (chemin configurable), contenant les informations récupérées des fichiers de logs.

7.2.1. Configuration

Exemple de fichier de configuration :

```
<jxla>
<dns>
<available>true</available>
<filecache>/home/logs/dns.cache</filecache>
</dns>

<logfiles>
<directory>/var/log/nvdcms/</directory>
<filenameregexp>http\.log(.)*</filenameregexp>
<format>
<regexp>m d h * * y $host $status $remote_ip "* $uri *"
* $agent $lang $referer *</regexp>
<regexp>m d h * * y $host $status $remote_ip "* $uri *"
* $lang $referer "*" * *</regexp>
<regexp>m d h * * y $host $status $remote_ip "* $uri *"
$agent $lang $referer "$user" "*" $size *</regexp>
</format>
</logfiles>

<pages>
<extensions>.xml, .htm, .html, .php, .php, .php4, .pl, .cgi</extensions>
</pages>

<max-values>
<referers>50</referers>
<keywords>50</keywords>
<user-agent>10</user-agent>
<remote-hosts>30</remote-hosts>
<uri>30</uri>
<notfound>20</notfound>
```

```
</max-values>

<localconfigclass>org.novadeck.jxla.config.EverydeckSiteConfig</localconfigclass>

<summary-name>summary.xml</summary-name>


<history-filename>/home/logs/logs.hist</history-filename>
<searchengines>
<engine>
<name>Yahoo!</name>
<requestparameter>p</requestparameter>
<domain>yahoo.fr</domain>
</engine>
</searchengines>

</jxla>
```

7.2.2. Démarrage

Pour pouvoir exécuter JXLA, il faut un compte utilisateur comme wwwrun qui a les droits de lecture sur le répertoire de logs (généralement "/var/log/nvdcms/") et les droits d'écriture dans le répertoire des serveurs web (généralement "/www").

L'exécution s'effectue à l'aide de la commande Java :

```
java org.novadeck.jxla.Main fichierDeConf.xml
```

ou du shell script (généralement situé dans le répertoire "/home/logs/") :

```
/home/logs/logs.sh
```

Sous Unix le script exécuté de façon périodique à l'aide d'un cron (éditer la crontab à l'aide du compte utilisateur "root" à l'aide de la commande "crontab -e -u wwwrun").

```
# Statistiques de fréquentation de sites
# (tous les jours à 1 heure du matin;)
0 1 * * *       /home/logs/logs.sh >> /home/logs/logs.out 2>&1
```

Chapitre 8. Répartition de charge

8.1. Configuration Linux LVS

Il est nécessaire d'ajouter les fonctionnalités LVS (http://www.linuxvirtualserver.org/) au noyau Linux pour lui permettre d'effectuer la répartition de charge (le noyau doit être patché et compilé avec les options LVS activées). Le logiciel *ipvsadm* doit être également installé avec le système.

Nous associons 2 adresses IP à l'interface externe (ici eth1).

```
#!/bin/sh
ifconfig eth1:0 x.x.x.100
ifconfig eth1:1 x.x.x.101
```

La configuration HTTP et HTTPS load balance l'ip x.x.x.100 sur 2 serveurs privés avec un poids de 1 sur le serveur 10.0.14.1 et 3 sur le serveur 10.0.14.2 (l'option -s wrr précise qu'il s'agit de l'argorithme de load balancing de type "weigh round robin", -w permet de spécifier le poids).

```
# HTTP
ipvsadm -A -t x.x.x.100:80 -s wrr
ipvsadm -a -t x.x.x.100:80 -r 10.0.0.1:8080 -m
ipvsadm -a -t x.x.x.100:80 -r 10.0.0.2:8080 -m -w 3

# HTTPS
ipvsadm -A -t x.x.x.100:443 -s wrr
ipvsadm -a -t x.x.x.100:443 -r 10.0.0.1:8443 -m
ipvsadm -a -t x.x.x.100:443 -r 10.0.0.2:8443 -m -w 3
```

Le load balancing DNS s'effectue des 2 adresses IP réelles vers 2 serveurs privés.

```
# DNS
ipvsadm -A -u x.x.x.100:53 -s rr
ipvsadm -a -u x.x.x.100:53 -r 10.0.1.1:53  -m
ipvsadm -A -u x.x.x.101:53 -s rr
ipvsadm -a -u x.x.x.101:53 -r 10.0.1.2:53  -m
```

La mise à zéro de la configuration LVS s'effectue de la façon suivante :

```
ipvsadm -C
ifconfig eth1:0 x.x.x.100 down
ifconfig eth1:1 x.x.x.101 down
```

8.2. Configuration Keepalive

Keepalive (http://keepalived.sourceforge.net/) fonctionne comme complément de LVS et assure le monitoring des services. En cas de défaillance d'un des services utilisés par LVS, keepalive informe le noyau Linux pour ajuster son fonctionnement.

Exemple de configuration keepalive reprenant la configuration LVS ci-dessus (uniquement pour le service HTTP).

```
# Configuration File for keepalived

global_defs {
notification_email {
admin@nvdcms.org
}
notification_email_from keepalive@nvdcms.org
smtp_server x.x.x.x
smtp_connect_timeout 30
lvs_id LVS_DEVEL
```

```
}

virtual_server x.x.x.100 80 {
delay_loop 10
lb_algo wrr
lb_kind NAT
#    persistence_timeout 10
protocol TCP

real_server 10.0.0.1 8080 {
weight 1
TCP_CHECK {
connect_timeout 10
nb_get_retry 300
delay_before_retry 5
}
}

real_server 10.0.0.2 8080 {
weight 3
TCP_CHECK {
connect_timeout 10
nb_get_retry 300
delay_before_retry 5
}
}
```