* help\_command

wls:/offline> help('all')

help('activate') Activate the changes.

help('addListener') Add a JMX listener to the specified MBean.

help('addTemplate') Extend the current domain.

help('adminHome') Administration MBeanHome.

help('assign') Assign resources to one or more destinations.

help('assignAll') (Deprecated) Assign all applications or services.

help('cancelEdit') Cancel an edit session.

help('cd') Navigate the hierarchy of beans.

help('closeDomain') Close the current domain.

help('closeTemplate') Close the current domain template.

help('closestore') Closes a store.

help('cmo') Current Management Object.

help('compactstore') Compacts and defragments the space occupied by a file store.

help('config') (Deprecated) Navigate to the last MBean in configuration hierarchy.

help('configToScript') Convert a domain configuration to WLST script.

help('connect') Connect WLST to a WebLogic Server instance.

help('connected') Variable indicating whether WLST is connected.

help('create') Create a configuration bean.

help('createDomain') Create a new domain.

help('currentTree') Return the current location in the hierarchy.

help('custom') Navigate to the root of custom MBeans.

help('delete') Delete a configuration bean.

help('deploy') Deploy an application.

help('disconnect') Disconnect WLST.

help('distributeApplication')Copy the deployment bundle to targets.

help('domainConfig') Navigate to last domain configuration MBean or root.

help('domainCustom') Navigate to the root of domain custom MBeans.

help('domainName') Name of the domain to which WLST is connected.

help('domainRuntime') Navigate to last domain runtime MBean or root.

help('domainRuntimeService')DomainRuntimeServiceMBean MBean.

help('dumpStack') Display stack trace from the last exception.

help('dumpVariables') Display all the variables used by WLST.

help('dumpstore') Dumps store contents in human-readable format to an XML file.

help('edit') Navigate to last edit configuration MBean or root.

< … list is much longer than printed here … >

wls:/offline> help('connect')

Description:

Connects WLST to a WebLogic Server instance.

You can specify the username and password on the command line, or you

can specify ran encrypted password that is stored locally by specifying the

locations of the user configuration and key files as arguments to the

connect command. For information about creating the user configuration

and key files, see help('storeUserConfig').

If you run the connect command without specifying the username and

<…>

wls:/mydomain/serverConfig>

wls:/offline> username = 'weblogic'

wls:/offline> password = 'weblogic'

wls:/offline> connect(username,password,'t3s://myhost:8001')

Connecting to weblogic server instance running at t3://localhost:8001 as

username weblogic...

Successfully connected to Admin Server 'AdminServer' that belongs to domain

'mydomain'.

wls:/mydomain/serverConfig>

wls:/offline> connect(userConfigFile='c:/myfiles/myuserconfigfile.secure', userKeyFile='c:/myfiles/myuserkeyfile.secure')

Connecting to weblogic server instance running at t3://localhost:7001

as username ...

Successfully connected to Admin Server 'AdminServer' that belongs to domain 'mydomain'.

wls:/mydomain/serverConfig>

wls:/offline>

* offline\_mode

Offline example:

wls:/offline> readTemplate('/opt/weblogic/10.3.6/wlserver\_10.3/common/templates/domains/wls.jar');

wls:/offline/base\_domain>pwd()

'/base\_domain'

wls:/offline/base\_domain>ls()

drw- Security

drw- Server

-rw- Active false

-rw- AdminServerName AdminServer

-rw- AdministrationMBeanAuditingEnabled false

-rw- AdministrationPort 9002

-rw- AdministrationPortEnabled false

-rw- AdministrationProtocol null

-rw- AutoDeployForSubmodulesEnabled true

-rw- ClusterConstraintsEnabled false

-rw- ConfigBackupEnabled false

-rw- ConfigurationAuditType null

-rw- ConfigurationVersion 10.3.6.0

-rw- ConsoleContextPath console

-rw- ConsoleEnabled true

-rw- ConsoleExtensionDirectory console-ext

-rw- DomainVersion 10.3.6.0

-rw- ExalogicOptimizationsEnabled false

-rw- GuardianEnabled false

-rw- InternalAppsDeployOnDemandEnabled true

-rw- LastModificationTime 0

-rw- MsgIdPrefixCompatibilityEnabled true

-rw- Name base\_domain

-rw- Notes null

-rw- OcmEnabled true

-rw- ProductionModeEnabled false

-rw- RootDirectory null

wls:/offline/base\_domain>cd ('Server')

wls:/offline/base\_domain/Server>ls()

drw- AdminServer

wls:/offline/base\_domain/Server>cmo

wls:/offline/base\_domain/Server>pwd()

'/base\_domain/Server'

wls:/offline/base\_domain/Server>

* online­\_mode

**wls:/offline>** connect('weblogic','test1234','t3://localhost:7001')

Connecting to t3://localhost:7001 with userid weblogic ...

Successfully connected to Admin Server 'AdminServer' that belongs to domain 'TestDomain'.

Warning: An insecure protocol was used to connect to the

server. To ensure on-the-wire security, the SSL port or

Admin port should be used instead.

**wls:/TestDomain/serverConfig**> cd ('Servers')

wls:/TestDomain/serverConfig/Servers> ls()

dr-- AdminServer

wls:/TestDomain/serverConfig/Servers> cd ('AdminServer')

wls:/TestDomain/serverConfig/Servers/AdminServer> pwd()

'serverConfig:/Servers/AdminServer'

wls:/TestDomain/serverConfig/Servers/AdminServer> ls()

dr-- COM

dr-- CandidateMachines

dr-- Cluster

dr-- CoherenceClusterSystemResource

dr-- DataSource

dr-- DefaultFileStore

dr-- ExecuteQueues

dr-- FederationServices

dr-- IIOP

dr-- JTAMigratableTarget

dr-- Log

dr-- Machine

dr-- NetworkAccessPoints

dr-- OverloadProtection

dr-- ReliableDeliveryPolicy

dr-- SSL

dr-- ServerDebug

dr-- ServerDiagnosticConfig

dr-- ServerStart

dr-- SingleSignOnServices

dr-- TransactionLogJDBCStore

dr-- WebServer

dr-- WebService

dr-- XMLEntityCache

dr-- XMLRegistry

-r-- AcceptBacklog 300

-r-- AddWorkManagerThreadsByCpuCount false

-r-- AdminReconnectIntervalSeconds 10

-r-- AdministrationPort 9002

-r-- AdministrationProtocol t3s

-r-- AutoKillIfFailed false

-r-- AutoMigrationEnabled false

-r-- AutoRestart true

-r-- COMEnabled false

-r-- ClasspathServletDisabled false

-r-- ClientCertProxyEnabled false

-r-- Cluster null

-r-- ClusterRuntime null

-r-- ClusterWeight 100

-r-- CoherenceClusterSystemResource null

-r-- CompleteCOMMessageTimeout -1

-r-- CompleteHTTPMessageTimeout -1

-r-- CompleteIIOPMessageTimeout -1

-r-- CompleteMessageTimeout 60

-r-- CompleteT3MessageTimeout -1

-r-- ConnectTimeout 0

-r-- CustomIdentityKeyStoreFileName null

-r-- CustomIdentityKeyStorePassPhrase \*\*\*\*\*\*

-r-- CustomIdentityKeyStorePassPhraseEncrypted \*\*\*\*\*\*

-r-- CustomIdentityKeyStoreType null

-r-- CustomTrustKeyStoreFileName null

-r-- CustomTrustKeyStorePassPhrase \*\*\*\*\*\*

-r-- CustomTrustKeyStorePassPhraseEncrypted \*\*\*\*\*\*

-r-- CustomTrustKeyStoreType null

-r-- DGCIdlePeriodsUntilTimeout 5

-r-- DefaultIIOPPassword \*\*\*\*\*\*

-r-- DefaultIIOPPasswordEncrypted \*\*\*\*\*\*

-r-- DefaultIIOPUser null

-r-- DefaultInternalServletsDisabled false

-r-- DefaultProtocol t3

-r-- DefaultSecureProtocol t3s

-r-- DefaultTGIOPPassword \*\*\*\*\*\*

-r-- DefaultTGIOPPasswordEncrypted \*\*\*\*\*\*

-r-- DefaultTGIOPUser guest

-r-- ExternalDNSName null

-r-- ExtraEjbcOptions null

-r-- ExtraRmicOptions null

-r-- GatheredWritesEnabled false

-r-- GracefulShutdownTimeout 0

-r-- HealthCheckIntervalSeconds 180

-r-- HealthCheckStartDelaySeconds 120

-r-- HealthCheckTimeoutSeconds 60

-r-- HostsMigratableServices true

-r-- HttpTraceSupportEnabled false

-r-- HttpdEnabled true

-r-- IIOPEnabled true

-r-- IIOPTxMechanism ots

-r-- IdleConnectionTimeout 65

-r-- IdleIIOPConnectionTimeout -1

-r-- IdlePeriodsUntilTimeout 4

-r-- IgnoreSessionsDuringShutdown false

-r-- InstrumentStackTraceEnabled true

-r-- InterfaceAddress null

-r-- JDBCLLRTableName null

-r-- JDBCLoggingEnabled false

-r-- JDBCLoginTimeoutSeconds 0

-r-- JMSDefaultConnectionFactoriesEnabled true

-r-- JNDITransportableObjectFactoryList null

-r-- JTAMigratableTarget null

-r-- JavaCompiler javac

-r-- JavaCompilerPostClassPath null

-r-- JavaCompilerPreClassPath null

-r-- JavaStandardTrustKeyStorePassPhrase \*\*\*\*\*\*

-r-- JavaStandardTrustKeyStorePassPhraseEncrypted \*\*\*\*\*\*

-r-- KeyStores DemoIdentityAndDemoTrust

-r-- ListenAddress 192.168.56.101

-r-- ListenDelaySecs 0

-r-- ListenPort 47001

-r-- ListenPortEnabled true

* available\_variables

wls:/MartinTest\_Domain/domainRuntime> dumpVariables()

cmgr [MBeanServerInvocationHandler]com.bea:Name=ConfigurationManager,Type=weblogic.management.mbeanservers.edit.ConfigurationManagerMBean

cmo [MBeanServerInvocationHandler]com.bea:Name=AppRuntimeStateRuntime,Type=AppRuntimeStateRuntime

connected true

domainName MartinTest\_Domain

domainRuntimeService [MBeanServerInvocationHandler]com.bea:Name=DomainRuntimeService,Type=weblogic.management.mbeanservers.domainruntime.DomainRuntimeServiceMBean

editService [MBeanServerInvocationHandler]com.bea:Name=EditService,Type=weblogic.management.mbeanservers.edit.EditServiceMBean

isAdminServer true

mbs javax.management.remote.rmi.RMIConnector$RemoteMBeanServerConnection@f55fa6e

recording false

runtimeService [MBeanServerInvocationHandler]com.bea:Name=RuntimeService,Type=weblogic.management.mbeanservers.runtime.RuntimeServiceMBean

scriptMode true

serverName AdminServer

typeService [MBeanServerInvocationHandler]com.bea:Name=MBeanTypeService,Type=weblogic.management.mbeanservers.MBeanTypeService

username weblogic

version WebLogic Server 10.3.5.0 Fri Apr 1 20:20:06 PDT 2011 1398638

exitonerror true

* mbean\_roots

wls:/MartinTest\_Domain/serverConfig>

wls:/MartinTest\_Domain/serverConfig> cmo

[MBeanServerInvocationHandler]com.bea:Name=MartinTest\_Domain,Type=Domain

wls:/MartinTest\_Domain/serverConfig> serverRuntime()

wls:/MartinTest\_Domain/serverRuntime> cmo

[MBeanServerInvocationHandler]com.bea:Name=AdminServer,Type=ServerRuntime

wls:/MartinTest\_Domain/serverRuntime> domainConfig()

wls:/MartinTest\_Domain/domainConfig> cmo

[MBeanServerInvocationHandler]com.bea:Name=MartinTest\_Domain,Location=MartinTest\_Domain,Type=Domain

wls:/MartinTest\_Domain/domainConfig> domainRuntime()

wls:/MartinTest\_Domain/domainRuntime> cmo

[MBeanServerInvocationHandler]com.bea:Name=MartinTest\_Domain,Type=DomainRuntime

wls:/MartinTest\_Domain/domainRuntime> custom()

Location changed to custom tree. This is a writable tree with No root.

For more help, use help(custom)

cmo

wls:/MartinTest\_Domain/custom> 'No Stub Available'

wls:/MartinTest\_Domain/custom> ls()

drw- JMImplementation

drw- com.sun.management

drw- com.sun.xml.ws.transport.http

drw- com.sun.xml.ws.util

drw- dbWLSMonitoring

drw- java.lang

drw- java.util.logging

wls:/MartinTest\_Domain/custom> domainCustom()

Location changed to domain custom tree. This is a writable tree with No root.

For more help, use help(domainCustom)

ls()

wls:/MartinTest\_Domain/domainCustom> drw- JMImplementation

drw- com.sun.management

drw- com.sun.xml.ws.transport.http

drw- com.sun.xml.ws.util

drw- dbWLSMonitoring

drw- java.lang

drw- java.util.logging

* start\_wlst

#!/bin/sh

MYDIR=`dirname $0`

MYFULLDIR=$(cd $MYDIR && pwd -P)

# set up WL\_HOME, the root directory of your WebLogic installation

#

# !!!!!!!!!!!!!!!! MUST BE ADAPTED TO INSTALLATION

WL\_HOME="/opt/oracle/wls/10.3.6/wlserver\_10.3"

JAVA\_HOME="/opt/jdks/jdk1.6"

umask 027

# set up common environment

. "${WL\_HOME}/server/bin/setWLSEnv.sh"

# all additional libraries like own mbeans, ... so that WLST knows about them.

#WLSTEXTENSIONCLASSPATH=${MYFULLDIR}/../libs/myOwnMBeans.jar

WLSTEXTENSIONCLASSPATH=

CLASSPATH="${MYFULLDIR}/../libs/jline-0\_9\_5.jar:${WLSTEXTENSIONCLASSPATH}:${CLASSPATH}${CLASSPATHSEP}${FMWLAUNCH\_CLASSPATH}${CLASSPATHSEP}${DERBY\_CLASSPATH}${CLASSPATHSEP}${DERBY\_TOOLS}${CLASSPATHSEP}${POINTBASE\_CLASSPATH}${CLASSPATHSEP}${POINTBASE\_TOOLS}"

#WLST\_PROPERTIES="-Dweblogic.wlstHome='.' ${WLST\_PROPERTIES}"

#

#export WLST\_PROPERTIES

JVM\_ARGS="-cp ${CLASSPATH} -Dprod.props.file='${WL\_HOME}'/.product.properties ${WLST\_PROPERTIES} ${JVM\_D64} -Xms256m -Xmx1024m -XX:PermSize=128m ${CONFIG\_JVM\_ARGS}"

eval '"${JAVA\_HOME}/bin/java"' ${JVM\_ARGS} jline.ConsoleRunner weblogic.WLST -skipWLSModuleScanning $1 $2 $3 $4 $5 $6 $7 $8

* easeSyntax

wls:/offline> connect('weblogic','test1234','t3://localhost:7001')

Connecting to t3://localhost:7001 with userid weblogic ...

Successfully connected to Admin Server 'AdminServer' that belongs to domain 'MartinTest\_Domain'.

Warning: An insecure protocol was used to connect to the

server. To ensure on-the-wire security, the SSL port or

Admin port should be used instead.

wls:/MartinTest\_Domain/serverConfig> easeSyntax()

You have chosen to ease syntax for some WLST commands.

However, the easy syntax should be strictly used in

interactive mode. Easy syntax will not function properly in

script mode and when used in loops. You can still use the

regular jython syntax although you have opted for easy

syntax.

Use easeSyntax to turn this off.

Use help(easeSyntax) for commands that support easy syntax

wls:/MartinTest\_Domain/serverConfig> cd Servers

wls:/MartinTest\_Domain/serverConfig/Servers> pwd

'serverConfig:/Servers'

wls:/MartinTest\_Domain/serverConfig/Servers> cmo

[MBeanServerInvocationHandler]com.bea:Name=MartinTest\_Domain,Type=Domain

wls:/MartinTest\_Domain/serverConfig/Servers> help

WLST is a command line scripting tool to configure and administer WebLogic Server. Try:

help('all') List all WLST commands available.

help('browse') List commands for browsing the hierarchy.

help('common') List the most commonly used commands.

help('control') List commands for controlling the domain/server.

help('deployment') List commands for deploying applications.

help('diagnostics') List commands for performing diagnostics.

help('editing') List commands for editing the configuration.

help('information') List commands for displaying information.

help('lifecycle') List commands for managing life cycle.

help('nodemanager') List commands for using Node Manager.

help('offline') List all offline commands available.

help('online') List all online commands available.

help('storeadmin') List all store admin commands.

help('trees') List commands use to navigate MBean hierarchy.

help('variables') List all global variables available.

* embedded\_wlst

package wlst;

import java.util.\*;

import weblogic.management.scripting.utils.WLSTInterpreter;

import org.python.util.InteractiveInterpreter;

public class EmbeddedWLST2 {

static InteractiveInterpreter interpreter = null;

EmbeddedWLST2()

{

System.out.println("EmbeddedWLST2...");

interpreter = new WLSTInterpreter();

}

private static void connect()

{

interpreter.execfile("MonitorThreads.py");

}

public static void main(String[] args)

{

System.out.println("main...");

new EmbeddedWLST2();

connect();

}

}

* create\_simple\_domain

###################################################################

# Create a very simple domain with only one admin server !!

###################################################################

print 'Creating the domain...'

domainsDirectory='/mydomains';

domainName = 'TestDomain';

readTemplate('/opt/wls/wlserver\_10.3/common/templates/domains/wls.jar');

# Setting listen address/port

cd('/Server/AdminServer')

set('ListenAddress','localhost');

set('ListenPort',7001);

# SSL Settings

create('AdminServer','SSL')

cd('SSL/AdminServer')

set('Enabled', 'true');

set('ListenPort',47002);

# Setting the username/password

cd('/Security/base\_domain/User/weblogic');

cmo.setName('weblogic');

cmo.setPassword('test1234');

# Set some important domain options

setOption('CreateStartMenu', 'false');

setOption('ServerStartMode', 'prod');

setOption('JavaHome', '/usr/lib/jvm/jre-1.6.0-openjdk.x86\_64');

setOption('OverwriteDomain', 'false');

print 'Writing Domain: '+ domainsDirectory+'/'+domainName;

writeDomain( domainsDirectory+'/'+domainName);

closeTemplate();

print 'Domain Created';

connUri = 't3://localhost:7001';

print 'Starting the Admin Server to test system ...';

startServer('AdminServer', domainName , connUri, 'weblogic', 'test1234', domainsDirectory+'/'+domainName, 'true', 60000, 'false');

print 'Started the Admin Server';

# Connecting to the Admin Server

connect('weblogic','test1234',connUri);

print 'Connected';

print 'Shutting down the Admin Server...';

shutdown();

print 'Exiting...';

exit();

* start/test\_adminserver

Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

Creating the domain...

Writing Domain: /domains/TestDomain

Domain Created

Starting the Admin Server to test system ...

Starting weblogic server ...

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:04 AM CET> <Info> <Security> <BEA-090905> <Disabling CryptoJ JCE Provider self-integrity check for better startup performance. To enable this check, specify -Dweblogic.security.allowCryptoJDefaultJCEVerification=true>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:04 AM CET> <Info> <Security> <BEA-090906> <Changing the default Random Number Generator in RSA CryptoJ from ECDRBG to FIPS186PRNG. To disable this change, specify -Dweblogic.security.allowCryptoJDefaultPRNG=true>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:04 AM CET> <Info> <WebLogicServer> <BEA-000377> <Starting WebLogic Server with OpenJDK 64-Bit Server VM Version 20.0-b11 from Sun Microsystems Inc.>

[...]

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:22 AM CET> <Notice> <Server> <BEA-002613> <Channel "Default" is now listening on 127.0.0.1:47001 for protocols iiop, t3, ldap, snmp, http.>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:22 AM CET> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure" is now listening on 127.0.0.1:47002 for protocols iiops, t3s, ldaps, https.>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:22 AM CET> <Notice> <WebLogicServer> <BEA-000329> <Started WebLogic Admin Server "AdminServer" for domain "TestDomain" running in Production Mode>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:22 AM CET> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:22 AM CET> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>

Server started successfully.

Started the Admin Server

Connecting to t3://localhost:47001 with userid weblogic ...

Successfully connected to Admin Server 'AdminServer' that belongs to domain 'TestDomain'.

Warning: An insecure protocol was used to connect to the

server. To ensure on-the-wire security, the SSL port or

Admin port should be used instead.

Connected

Shutting down the Admin Server...

Shutting down the server AdminServer with force=false while connected to AdminServer ...

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Alert> <WebLogicServer> <BEA-000396> <Server shutdown has been requested by weblogic>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to SUSPENDING>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <HTTP> <BEA-101278> <There are no active sessions. The Web service is ready to suspend.>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to ADMIN>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to SHUTTING\_DOWN>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <Server> <BEA-002607> <Channel "DefaultSecure" listening on 127.0.0.1:47002 was shutdown.>

WLST-WLS-1352361843103: <Nov 8, 2012 9:04:40 AM CET> <Notice> <Server> <BEA-002607> <Channel "Default" listening on 127.0.0.1:47001 was shutdown.>

WLST-WLS-1352361843103: Stopped draining WLST-WLS-1352361843103

WLST-WLS-1352361843103: Stopped draining WLST-WLS-1352361843103

WLST lost connection to the WebLogic Server that you were

connected to, this may happen if the server was shutdown or

partitioned. You will have to re-connect to the server once the

server is available.

Disconnected from weblogic server: AdminServer

Disconnected from weblogic server:

Exiting...

* create\_simple\_domain\_(extended)

###################################################################

# Create a very simple domain with only one admin server !!

###################################################################

from java.io import FileInputStream

from java.io import File

print 'Creating the domain...'

domainsDirectory='/user\_data/weblogic\_domains';

domainName = 'TestDomain';

readTemplate('/opt/wls/wlserver\_10.3/common/templates/domains/wls.jar');

# Setting listen address/port

cd('/Server/AdminServer')

set('ListenAddress','localhost');

set('ListenPort',47001);

# SSL Settings

create('AdminServer','SSL')

cd('SSL/AdminServer')

set('Enabled', 'true');

set('ListenPort',47002);

# Setting the username/password

cd('/Security/'+ domainName +'/User/weblogic');

cmo.setName('weblogic');

cmo.setPassword('test1234');

setOption('CreateStartMenu', 'false');

setOption('ServerStartMode', 'prod');

setOption('JavaHome', '/usr/lib/jvm/jre-1.6.0-openjdk.x86\_64');

setOption('OverwriteDomain', 'false');

print 'Writing Domain: '+ domainsDirectory+'/'+domainName;

writeDomain( domainsDirectory+'/'+domainName);

closeTemplate();

print 'Domain Created';

***# Create boot.properties file !***

***os.makedirs(domainsDirectory+"/"+domainName+ "/servers/AdminServer/security")***

***f=open(domainsDirectory+"/"+domainName + "/servers/AdminServer/security/boot.properties" , 'w')***

***f.write('username=weblogic')***

***f.write('password=test1234')***

***f.flush()***

***f.close()***

connUri = 't3://localhost:47001';

print 'Starting the Admin Server to test system ...';

startServer('AdminServer', domainName , connUri, 'weblogic', 'test1234', domainsDirectory+'/'+domainName,'true',60000,'false');

print 'Started the Admin Server';

# Connecting to the Admin Server

connect('weblogic','test1234',connUri);

print 'Connected';

print 'Shutting down the Admin Server...';

shutdown();

print 'Exiting...';

exit();

* create\_server\_template/cluster

# change to the root folder

cd('/')

# create a new server template

cmo.createServerTemplate('MyDynServer\_Template')

# change to the new server template

cd('/ServerTemplates/MyDynServer\_Template')

cmo.setListenPort(6000)

# change the SSL port

cd('/ServerTemplates/MyDynServer\_Template/SSL/MyDynServer\_Template')

cmo.setListenPort(6100)

# set a machine – None means that the adminserver can target the new instances dynamically

cd('/ServerTemplates/MyDynServer\_Template')

cmo.setMachine(None)

# create a new dynamic cluster

cd('/')

cmo.createCluster('Cluster-1234')

# configure the cluster

cd('/Clusters/Cluster-1234')

cmo.setClusterMessagingMode('unicast')

# set the template which should be used to create server instances

cd('/ServerTemplates/MyDynServer\_Template')

cmo.setCluster(getMBean('/Clusters/Cluster-1234'))

# configure the cluster

cd('/Clusters/Cluster-1234/DynamicServers/Cluster-1234')

cmo.setServerTemplate(getMBean('/ServerTemplates/MyDynServer\_Template'))

cmo.setMaximumDynamicServerCount(3)

cmo.setCalculatedListenPorts(true)

cmo.setCalculatedMachineNames(true)

cmo.setCalculatedListenPorts(true)

cmo.setServerNamePrefix('MyDynServer\_')

# activate the changes

activate()

* migration

cd('/')

# Create a first managed server

cmo.createServer('TestServer1')

cd('/Servers/TestServer1')

cmo.setListenAddress('')

cmo.setListenPort(7001)

cd('/')

# create a cluster

cmo.createCluster('TestCluster')

# configure cluster

cd('/Clusters/TestCluster')

cmo.setClusterMessagingMode('unicast')

cmo.setClusterBroadcastChannel('')

# Assign the first server to the cluster

cd('/Servers/TestServer1')

cmo.setCluster(getMBean('/Clusters/TestCluster'))

cd('/')

# Create a second managed server

cmo.createServer('TestServer2')

cd('/Servers/TestServer2')

cmo.setListenAddress('')

cmo.setListenPort(7001)

# assign the second server also to our testcluster

cmo.setCluster(getMBean('/Clusters/TestCluster'))

cd('/Servers/TestServer1')

cmo.setListenPortEnabled(true)

cmo.setJavaCompiler('javac')

cmo.setClientCertProxyEnabled(false)

cmo.setMachine(getMBean('/Machines/localhost'))

cd('/Servers/TestServer2')

cmo.setListenPortEnabled(true)

cmo.setJavaCompiler('javac')

cmo.setClientCertProxyEnabled(false)

cmo.setMachine(getMBean('/Machines/test12345'))

cd('/Servers/TestServer1')

# set auto migration for the first server to false

cmo.setAutoMigrationEnabled(false)

# set all possible machines (including own !) to which this service might be migrated

set('CandidateMachines',jarray.array([ObjectName('com.bea:Name=localhost,Type=Machine'), ObjectName('com.bea:Name=test12345,Type=UnixMachine')], ObjectName))

cd('/MigratableTargets/TestServer1 (migratable)')

# set all possible migration server (including own !) to which this service might be migrated

set('ConstrainedCandidateServers',jarray.array([ObjectName('com.bea:Name=TestServer2,Type=Server'), ObjectName('com.bea:Name=TestServer1,Type=Server')], ObjectName))

cd('/Servers/TestServer1/JTAMigratableTarget/TestServer1')

# set all possible JTA migration targets (including own !) to which this service might be migrated

set('ConstrainedCandidateServers',jarray.array([ObjectName('com.bea:Name=TestServer2,Type=Server'), ObjectName('com.bea:Name=TestServer1,Type=Server')], ObjectName))

cmo.setMigrationPolicy('manual')

cd('/Servers/TestServer2')

# set auto migration for the second server to false

cmo.setAutoMigrationEnabled(false)

# set all possible machines (including own !) to which the second service might be migrated

set('CandidateMachines',jarray.array([ObjectName('com.bea:Name=localhost,Type=Machine')], ObjectName))

# set all possible migration server (including own !) to which this service might be migrated

cd('/MigratableTargets/TestServer2 (migratable)')

set('ConstrainedCandidateServers',jarray.array([ObjectName('com.bea:Name=TestServer1,Type=Server'), ObjectName('com.bea:Name=TestServer2,Type=Server')], ObjectName))

# set all possible JTA migration targets (including own !) to which the second service might be migrated

cd('/Servers/TestServer2/JTAMigratableTarget/TestServer2')

set('ConstrainedCandidateServers',jarray.array([ObjectName('com.bea:Name=TestServer1,Type=Server'), ObjectName('com.bea:Name=TestServer2,Type=Server')], ObjectName))

cmo.setMigrationPolicy('manual')

* set\_JTA\_timeout

###################################################################

# Setting Domain JTA Transaction timeout

###################################################################

def setDomainJTATimeout():

try:

print 'Setting Domain JTA Transaction timeout...';

edit();

startEdit();

cd('/JTA/'+domainProps.getProperty('domainName'));

# Maximum amount of time, in seconds, an active transaction

# is allowed to be in the first phase of a transaction

cmo.setTimeoutSeconds(300);

save();

activate();

except:

print 'Exception while setting Domain JTA Transaction timeout !';

dumpStack();

exit();

* set\_JTA\_properties

###################################################################

# Setting Domain JTA Transaction properties

###################################################################

def setDomainJTAProperties():

try:

print 'Setting Domain JTA Transaction properties ...';

edit();

startEdit();

cd('/JTA/'+domainProps.getProperty('domainName'));

# Maximum time, an active transaction is allowed to be in the first phase of a transaction

cmo.setTimeoutSeconds(300);

# The maximum number of simultaneous in-progress transactions allowed

cmo.setMaxTransactions(20000)

# The time a transaction manager waits for transactions involving the resource to complete

cmo.setUnregisterResourceGracePeriod(25)

# maximum time a transaction manager persists in attempting to complete the second phase

cmo.setAbandonTimeoutSeconds(80000)

# Indicates that XA calls are executed in parallel if there are available threads

cmo.setParallelXAEnabled(true)

# automatically performs an XA Resource forget for heuristic transaction completions

cmo.setForgetHeuristics(true)

# the two-phase commit protocol is used

cmo.setTwoPhaseEnabled(true)

# maximum cycles that the transaction manager performs the beforeCompletion synchronization

cmo.setBeforeCompletionIterationLimit(20)

# interval the transaction manager creates a new transaction log

cmo.setCheckpointIntervalSeconds(200)

# Specifies transport security mode required by WebService Transaction endpoints

cmo.setSecurityInteropMode('default')

# XA calls are executed in parallel if there are available threads

cmo.setParallelXAEnabled(false)

# Maximum number of concurrent requests to resources allowed for each server

cmo.setMaxResourceRequestsOnServer(60)

# transport security mode required by WebService Transaction endpoints

cmo.setWSATTransportSecurityMode('SSLNotRequired')

# Maximum allowed time duration, in milliseconds, for XA calls to resources

cmo.setMaxXACallMillis(100000)

# maximum time, in seconds, a transaction manager waits for all resource managers to respond

cmo.setCompletionTimeoutSeconds(0)

# Maximum duration time, in milliseconds, that a resource is declared dead

cmo.setMaxResourceUnavailableMillis(1500000)

save();

activate();

except:

print 'Exception while setting Domain JTA Transaction timeout !';

dumpStack();

exit();

* configure\_startup\_class

# go into edit mode

edit()

# start edit

startEdit()

# test if the startup class is already configured

# the idea is to get a reference to the startup class mbean. If this mbean does not exit then this

# startup class has not yet been configured in this server and we can create it

cd('/')

startup\_class = getMBean('/StartupClasses/TestStartupClass')

if startup\_class == None:

cmo.createStartupClass('TestStartupClass')

# now we can be sure that the startup calss either was already there or has been created

cd('/StartupClasses/TestStartupClass')

# configure classname

cmo.setClassName('com.wlsmonitoringbook.teststartup')

# set the target(s) for this startuo class

set('Targets',jarray.array([ObjectName('com.bea:Name=AdminServer,Type=Server'), ObjectName('com.bea:Name=Messaging\_Cluster,Type=Cluster')], ObjectName))

# define when the class is to be called

cmo.setLoadBeforeAppDeployments(true)

cmo.setLoadBeforeAppActivation(true)

cmo.setDeploymentOrder(1000)

cmo.setFailureIsFatal(true)

# further arguments which will be passed without modification into the init method of the startup

cmo.setArguments('initarg=xyz')

* configure\_shutdown\_class

cd('/')

cmo.createShutdownClass('TestShutdownClass')

cd('/Deployments/TestShutdownClass')

cmo.setClassName('com.wlsmonitoringbook.testshutdown')

set('Targets',jarray.array([ObjectName('com.bea:Name=BusinessTier\_Cluster,Type=Cluster'), ObjectName('com.bea:Name=WebTier\_Cluster,Type=Cluster')], ObjectName))

cmo.setDeploymentOrder(1000)

* setup\_network\_channels

# switch to the managed server

cd('/Servers/MartinTest\_Domain\_MS2')

# create the DMZ network

cmo.createNetworkAccessPoint('DMZ-Network')

# switch to the DMZ network channel

cd('/Servers/MartinTest\_Domain\_MS2/NetworkAccessPoints/DMZ-Network')

# set the protocol, the DMZ network address and port

cmo.setProtocol('t3')

cmo.setListenAddress('192.168.56.26')

cmo.setListenPort(20001)

cmo.setEnabled(true)

# configure channel properties

cmo.setHttpEnabledForThisProtocol(false)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

# switch back to the server

cd('/Servers/MartinTest\_Domain\_MS2')

# create the INTERNET channel

cmo.createNetworkAccessPoint('INTERNET')

# switch to the internet channel and configure the https channel

cd('/Servers/MartinTest\_Domain\_MS2/NetworkAccessPoints/INTERNET')

cmo.setProtocol('https')

cmo.setListenAddress('192.168.60.112')

cmo.setListenPort(20002)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

# configure https properties

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(true)

cmo.setClientCertificateEnforced(false)

# again back to the server

cd('/Servers/MartinTest\_Domain\_MS2')

# create the intranet http access channel

cmo.createNetworkAccessPoint('Intranet\_http')

# switch to the channel and configure the http protocol, the correct network address and port

cd('/Servers/MartinTest\_Domain\_MS2/NetworkAccessPoints/Intranet\_http')

cmo.setProtocol('http')

cmo.setListenAddress('192.168.100.215')

cmo.setListenPort(20003)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

# last time back to the server

cd('/Servers/MartinTest\_Domain\_MS2')

# create the access channel for CORBA based applications (like batch clients)

cmo.createNetworkAccessPoint('Intranet\_IIOP')

# chage to the IIOP channel and configure again the correct network and port

cd('/Servers/MartinTest\_Domain\_MS2/NetworkAccessPoints/Intranet\_IIOP')

cmo.setProtocol('iiop')

cmo.setListenAddress('192.168.100.215')

cmo.setListenPort(30003)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(false)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

#############################################################################

# Now do the same for the other managed-server

cd('/Servers/MartinTest\_Domain\_MS1')

cmo.createNetworkAccessPoint('DMZ-Network')

cd('/Servers/MartinTest\_Domain\_MS1/NetworkAccessPoints/DMZ-Network')

cmo.setProtocol('t3')

cmo.setListenAddress('192.168.56.25')

cmo.setListenPort(20001)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

cd('/Servers/MartinTest\_Domain\_MS1')

cmo.createNetworkAccessPoint('INTERNET')

cd('/Servers/MartinTest\_Domain\_MS1/NetworkAccessPoints/INTERNET')

cmo.setProtocol('https')

cmo.setListenAddress('192.168.60.111')

cmo.setListenPort(20002)

cmo.setPublicPort(40010)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

cmo.setPublicPort(20002)

cd('/Servers/MartinTest\_Domain\_MS1')

cmo.createNetworkAccessPoint('Intranet\_http')

cd('/Servers/MartinTest\_Domain\_MS1/NetworkAccessPoints/Intranet\_http')

cmo.setProtocol('http')

cmo.setListenAddress('192.168.100.214')

cmo.setListenPort(20003)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

cd('/Servers/MartinTest\_Domain\_MS1')

cmo.createNetworkAccessPoint('Intranet\_IIOP')

cd('/Servers/MartinTest\_Domain\_MS1/NetworkAccessPoints/Intranet\_IIOP')

cmo.setProtocol('iiop')

cmo.setListenAddress('192.168.100.214')

cmo.setListenPort(30003)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(false)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

* check\_session

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

pathSeparator = '/';

domainProps = Properties();

userConfigFile = '';

userKeyFile = '';

consoleAction = '';

###################################################################

# Load properties

###################################################################

def intialize():

global domainProps;

global userConfigFile;

global userKeyFile;

global consoleAction;

# test arguments

if len(sys.argv) != 3:

print 'Usage: checkforsession.sh <property\_file>';

exit();

try:

domainProps = Properties()

# load properties and overwrite defaults

input = FileInputStream(sys.argv[2])

domainProps.load(input)

input.close()

userConfigFile = File(domainProps.getProperty('userconfig')

userKeyFile = File(domainProps.getProperty('userkey')

except:

print 'Cannot load properties !';

exit();

print 'Initialization completed';

###################################################################

# check if session is active

###################################################################

def existActiveEditSession():

# check for existing sessions using the configuration manager

myConfigurationManager = getConfigManager()

# try to get an active session.

# if call succeeds, then a session is available, otherwise an exception is thrown

try:

# Test if an active session is available and if yes return it

myConfigurationManager.getChanges()

print 'Active edit session found !'

# test which user is making changes

userWhoOwnsCurrentEditSession = myConfigurationManager.getCurrentEditor()

print ' - The active session belongs to ' + userWhoOwnsCurrentEditSession

# out of interst, return the number of active changes in the current session

changeList = myConfigurationManager.getUnactivatedChanges()

numberOfUnactivatedChanges = len(changeList)

print ' - The actual session has '+str(numberOfUnactivatedChanges)+

' not yet activated changes '

print '\n Waiting list of changes:'

if (numberOfUnactivatedChanges > 0):

for nextChange in changeList:

print nextChange

print '\n'

return 'true'

except:

dumpStack()

# good for us - no active changes found

print 'No active edit session found !'

return 'false'

###################################################################

# Main Code Execution

###################################################################

if \_\_name\_\_== "main":

intialize()

connUri = domainProps.getProperty('adminserverURL)

print 'Connecting to the Admin Server ('+connUri+')';

connect(userConfigFile=userConfigFile,userKeyFile=userKeyFile,url=connUri);

if ('true' == existActiveEditSession()):

print '\n OH OH session already exists\n'

else:

print '\n Good news - session does not yet exist !\n'

print 'Disconnect from the Admin Server...';

disconnect();

* create\_non\_XA\_datasource

cd('/')

# using the create operation on root level to create a new datasource

cmo.createJDBCSystemResource('NonXA\_TestDatasource')

# change to the datasource mbean

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource')

# set the name

cmo.setName('NonXA\_TestDatasource')

# change to the "JDBCDataSourceParams" mbean of the new datasource and set the JNDI name

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCDataSourceParams/NonXA\_TestDatasource')

set('JNDINames',jarray.array([String('jndi/nonXAtest')], String))

# change to the "JDBCDriverParams" mbean of the new datasource and set the connection details like url , driver and password

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCDriverParams/NonXA\_TestDatasource')

cmo.setUrl('jdbc:oracle:thin:@mydbhost.com:1521:TestDB')

cmo.setDriverName('oracle.jdbc.OracleDriver')

cmo.setPassword('db\_password');

# change to the JDBCConnectionPoolParams mbean and set connection specific details

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCConnectionPoolParams/NonXA\_TestDatasource')

cmo.setTestTableName('SQL SELECT 1 FROM DUAL\r\n')

# creating a property for the user name

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCDriverParams/NonXA\_TestDatasource/Properties/NonXA\_TestDatasource')

cmo.createProperty('user')

# set the user name

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCDriverParams/NonXA\_TestDatasource/Properties/NonXA\_TestDatasource/Properties/user')

cmo.setValue('dbuser')

# setting transactional options

cd('/JDBCSystemResources/NonXA\_TestDatasource/JDBCResource/NonXA\_TestDatasource/JDBCDataSourceParams/NonXA\_TestDatasource')

cmo.setGlobalTransactionsProtocol('OnePhaseCommit')

# Finally target the datasource - in this case to a cluster

cd('/SystemResources/NonXA\_TestDatasource')

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')], ObjectName))

* 2\_XA\_example

cd('/')

cmo.createJDBCSystemResource('XA\_TestDatasource')

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource')

cmo.setName('XA\_TestDatasource')

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCDataSourceParams/XA\_TestDatasource')

set('JNDINames',jarray.array([String('jndi/xaTest')], String))

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCDriverParams/XA\_TestDatasource')

cmo.setUrl('jdbc:oracle:thin:@myxadbhost.com:1521:TestXADB')

cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')

cmo.setPassword('db2\_password');

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCConnectionPoolParams/XA\_TestDatasource')

cmo.setTestTableName('SQL SELECT 1 FROM DUAL\r\n')

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCDriverParams/XA\_TestDatasource/Properties/XA\_TestDatasource')

cmo.createProperty('user')

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCDriverParams/XA\_TestDatasource/Properties/XA\_TestDatasource/Properties/user')

cmo.setValue('dbuser')

cd('/JDBCSystemResources/XA\_TestDatasource/JDBCResource/XA\_TestDatasource/JDBCDataSourceParams/XA\_TestDatasource')

cmo.setGlobalTransactionsProtocol('TwoPhaseCommit')

cd('/SystemResources/XA\_TestDatasource')

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')], ObjectName))

cd('/')

cmo.createJDBCSystemResource('XA\_TestDatasource2')

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2')

cmo.setName('XA\_TestDatasource2')

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCDataSourceParams/XA\_TestDatasource2')

set('JNDINames',jarray.array([String('jndi/xa\_test2')], String))

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCDriverParams/XA\_TestDatasource2')

cmo.setUrl('jdbc:oracle:thin:@myxa2dbhost.com:1521:Test2XADB')

cmo.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')

cmo.setPassword('db3\_password');

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCConnectionPoolParams/XA\_TestDatasource2')

cmo.setTestTableName('SQL SELECT 1 FROM DUAL\r\n\r\n')

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCDriverParams/XA\_TestDatasource2/Properties/XA\_TestDatasource2')

cmo.createProperty('user')

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCDriverParams/XA\_TestDatasource2/Properties/XA\_TestDatasource2/Properties/user')

cmo.setValue('dbuser')

cd('/JDBCSystemResources/XA\_TestDatasource2/JDBCResource/XA\_TestDatasource2/JDBCDataSourceParams/XA\_TestDatasource2')

cmo.setGlobalTransactionsProtocol('TwoPhaseCommit')

cd('/SystemResources/XA\_TestDatasource2')

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')], ObjectName))

* configure\_datasources

###################################################################

# create all datasources

###################################################################

def createAllDatasources():

try:

totalDataSource\_to\_Create=domainProps.get("amountDatasources")

edit()

startEdit()

print 'Creating All DataSources ....'

i=1

while (i <= int(totalDataSource\_to\_Create)) :

try:

cd('/')

datasource\_name = get\_instance\_property('datasource',str(i), 'name');

datasource\_targettype = get\_instance\_property('datasource',str(i), 'targettype');

datasource\_target = get\_instance\_property('datasource',str(i), 'target');

datasource\_jndiname = get\_instance\_property('datasource',str(i), 'jndiname');

datasource\_driver\_class =

get\_instance\_property('datasource',str(i), 'driver\_class');

datasource\_url = get\_instance\_property('datasource',str(i), 'url');

datasource\_username = get\_instance\_property('datasource',str(i), 'username');

datasource\_password = get\_instance\_property('datasource',str(i), 'password');

datasource\_maxcapacity = get\_instance\_property('datasource',str(i), 'maxcapacity');

datasource\_testquery = get\_instance\_property('datasource',str(i), 'testquery');

# GlobalTransactionsProtocol ONLY for NON-XA datasources possible !!!!

# values possible: 'None' and 'OnePhaseCommit' for XA: 'TwoPhaseCommit'

datasource\_globalTransactionsProtocol =

get\_instance\_property('datasource',str(i), 'globalTransactionsProtocol');

# Creating DataSource

cmo.createJDBCSystemResource(datasource\_name)

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name)

cmo.setName(datasource\_name)

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDataSourceParams/' + datasource\_name )

set('JNDINames',jarray.array([String(datasource\_jndiname)], String))

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name )

cmo.setUrl(datasource\_url)

cmo.setDriverName( datasource\_driver\_class );

cmo.setPassword(datasource\_password);

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCConnectionPoolParams/' + datasource\_name )

cmo.setTestTableName(datasource\_testquery);

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name + '/Properties/' + datasource\_name )

cmo.createProperty('user')

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name + '/Properties/' + datasource\_name

+ '/Properties/user')

cmo.setValue(datasource\_username);

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDataSourceParams/' + datasource\_name )

cmo.setGlobalTransactionsProtocol(datasource\_globalTransactionsProtocol);

cd('/SystemResources/' + datasource\_name );

set('Targets',jarray.array([ObjectName('com.bea:Name=' + datasource\_target

+ ',Type='+datasource\_targettype)], ObjectName))

print 'DataSource: ',datasource\_name,', has been created Successfully !!!'

except:

dumpStack();

print '\*\*\*\*\* CANNOT CREATE DATASOURCE !!! Check If the DataSource With the Name : ' +

datasource\_name +' Alreday exists or NOT...'

print ''

i = i + 1

save()

activate()

except:

print 'Exception while creating datasources - please check databases !';

dumpStack();

* datasources\_wallet

###################################################################

# create all datasources

###################################################################

def createAllDatasources():

try:

totalDataSource\_to\_Create=domainProps.get("amountDatasources")

edit()

startEdit()

print 'Creating All DataSources ....'

i=1

while (i <= int(totalDataSource\_to\_Create)) :

try:

datasource\_name = get\_instance\_property('datasource',str(i), 'name');

datasource\_targettype = get\_instance\_property('datasource',str(i), 'targettype');

datasource\_target = get\_instance\_property('datasource',str(i), 'target');

datasource\_jndiname = get\_instance\_property('datasource',str(i), 'jndiname');

datasource\_relativeWalletDir

= get\_instance\_property('datasource',str(i), 'relativeWalletDir');

datasource\_driver\_class = get\_instance\_property('datasource',str(i), 'driver\_class');

datasource\_url = get\_instance\_property('datasource',str(i), 'url');

datasource\_maxcapacity = get\_instance\_property('datasource',str(i), 'maxcapacity');

datasource\_testquery = get\_instance\_property('datasource',str(i), 'testquery');

datasource\_globalTransactionsProtocol = get\_instance\_property('datasource',str(i),

'globalTransactionsProtocol');

# Creating DataSource

cd('/')

cmo.createJDBCSystemResource(datasource\_name)

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name)

cmo.setName(datasource\_name)

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDataSourceParams/' + datasource\_name )

set('JNDINames',jarray.array([String(datasource\_jndiname)], String))

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name )

cmo.setUrl(datasource\_url)

cmo.setDriverName( datasource\_driver\_class );

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCConnectionPoolParams/' + datasource\_name )

cmo.setTestTableName(datasource\_testquery);

# WALLET

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name + '/Properties/' + datasource\_name )

cmo.createProperty('oracle.net.wallet\_location')

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDriverParams/' + datasource\_name + '/Properties/' + datasource\_name

+ '/Properties/oracle.net.wallet\_location')

cmo.setValue(domainProps.getProperty('walletsDirectory')+

'/'+datasource\_relativeWalletDir);

cd('/JDBCSystemResources/' + datasource\_name + '/JDBCResource/' + datasource\_name

+ '/JDBCDataSourceParams/' + datasource\_name )

cmo.setGlobalTransactionsProtocol(datasource\_globalTransactionsProtocol);

cd('/SystemResources/' + datasource\_name );

set('Targets',jarray.array([ObjectName('com.bea:Name=' + datasource\_target

+ ',Type='+datasource\_targettype)], ObjectName))

print 'DataSource: ',datasource\_name,', has been created Successfully !!!'

except:

dumpStack();

print '\*\*\*\*\* CANNOT CREATE DATASOURCE !!! Check If the DataSource With the Name : '

, datasource\_name ,' Alreday exists or NOT...'

print ''

i = i + 1

save()

activate()

except:

print 'Exception while creating datasources - please check databases !';

dumpStack();

* function\_definitions

# conncto to the admin server

connect('weblogic','xxxx','t3://localhost:7001')

try:

# start an edit session

edit()

startEdit()

cd('/')

# create different instances of JMS servers

createAnewJMSServer('MyTestServer\_1','mytest\_filestore\_1.store','AdminServer')

createAnewJMSServer('MyTestServer\_2','mytest\_filestore\_2.store','AdminServer')

createAnewJMSServer('MyTestServer\_3','mytest\_filestore\_3.store','AdminServer')

# create different JMS modules

createJMSModule('testModule\_1', 'Server', 'AdminServer')

createJMSModule('testModule\_2', 'Server', 'AdminServer')

createJMSModule('testModule\_3', 'Server', 'AdminServer')

# now create a number of connection factories on the different modules

createJmsConnectionFactory('testModule\_1', 'connection\_1\_1', 'jms/connection\_1\_1')

createJmsConnectionFactory('testModule\_1', 'connection\_1\_2', 'jms/connection\_1\_2')

createJmsConnectionFactory('testModule\_1', 'connection\_1\_3', 'jms/connection\_1\_3')

createJmsConnectionFactory('testModule\_2', 'connection\_2\_1', 'jms/connection\_2\_1')

createJmsConnectionFactory('testModule\_2', 'connection\_2\_2', 'jms/connection\_2\_2')

createJmsConnectionFactory('testModule\_3', 'connection\_3\_1', 'jms/connection\_3\_1')

createJmsConnectionFactory('testModule\_3', 'connection\_3\_2', 'jms/connection\_3\_2')

# example how to create a XA connection factory

createJms\_XA\_ConnectionFactory('testModule\_1', 'connection\_xa\_1\_1', 'jms/XA\_connection\_1\_1')

# create different sub deployments

createJMSSubDeployment('testModule\_1','subDeployment\_1', 'MyTestServer\_1')

createJMSSubDeployment('testModule\_2','subDeployment\_2', 'MyTestServer\_2')

createJMSSubDeployment('testModule\_3','subDeployment\_3', 'MyTestServer\_3')

# finally create QUEUE destinations

createQueue('testModule\_1', 'Queue\_1a', 'jms/Queue\_1a', 'subDeployment\_1')

createQueue('testModule\_1', 'Queue\_1b', 'jms/Queue\_1b', 'subDeployment\_1')

createQueue('testModule\_1', 'Queue\_1c', 'jms/Queue\_1c', 'subDeployment\_1')

createQueue('testModule\_2', 'Queue\_2', 'jms/Queue\_2', 'subDeployment\_2')

createQueue('testModule\_3', 'Queue\_3', 'jms/Queue\_3', 'subDeployment\_3')

# create a queue with error handling

createQueueWithErrorHandling('testModule\_1', 'Queue\_ERR\_1a',

'jms/Queue\_ERR\_1a', 'subDeployment\_1', 'Queue\_1c')

# create a TOPIC destination

createTopic('MyTestServer\_1', 'testModule\_1', 'Topic\_1a', 'jms/Topic\_1a', 'subDeployment\_1')

# save changes and activate

save()

activate()

except Exception, e:

print(e)

dumpStack()

disconnect()

* foreign\_JMS\_server\_configuration

def createForeignJMSServer(jmsModuleName,foreignName,foreignConnectionFactoryName,

cfnLocalName, cfnRemoteName,connectionURL):

print 'Start to create foreign JMS server: '+foreignName

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName)

cmo.createForeignServer(foreignName)

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+foreignName)

cmo.setDefaultTargetingEnabled(true)

cmo.createForeignConnectionFactory(foreignConnectionFactoryName)

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+

foreignName+'/ForeignConnectionFactories/'+foreignConnectionFactoryName)

cmo.setLocalJNDIName(cfnLocalName)

cmo.setRemoteJNDIName(cfnRemoteName)

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+foreignName)

cmo.setConnectionURL(connectionURL)

cmo.setInitialContextFactory('com.xyz.jndi.ctx.InitialContextFactoryWrapper')

cmo.unSet('JNDIPropertiesCredentialEncrypted')

cmo.createJNDIProperty('factory')

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+

foreignName+'/JNDIProperties/factory')

cmo.setValue('com.sun.jndi.fscontext.RefFSContextFactory')

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+foreignName)

cmo.createJNDIProperty('SECURITY\_AUTHENTICATION')

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+

foreignName+'/JNDIProperties/SECURITY\_AUTHENTICATION')

cmo.setValue('none')

# setup foreign destinations

print 'Start to create foreign JMS destinations for server: '+foreignName

ListJMS = ["foreign\_dest1","foreign\_dest2","foreign\_dest3","foreign\_dest4","foreign\_dest5"]

for idJms in range(0, len(ListJMS)):

cd('/JMSSystemResources/'+jmsModuleName+'/JMSResource/'+jmsModuleName+'/ForeignServers/'+foreignName)

localName = 'jms/' + ListJMS[idJms]

remoteName = ListJMS[idJms]

myFD = cmo.createForeignDestination(localName)

cd('ForeignDestinations')

myFD.setLocalJNDIName(localName)

myFD.setRemoteJNDIName(remoteName)

* JNDI\_provider

# connect to to the server

connect('weblogic','test1234','t3://localhost:7001')

# start an edit session

edit()

startEdit()

# if the JNDI provider does not exist, then create it

if getMBean("/ForeignJNDIProviders/myTestJNDIProvider") is None:

foreignJNDIInstance=create('myTestJNDIProvider',"ForeignJNDIProvider")

foreignJNDIInstance.addTarget(getMBean('/Clusters/WebTier\_Cluster'))

foreignJNDIInstance.setInitialContextFactory('weblogic.jndi.WLInitialContextFactory')

foreignJNDIInstance.setProviderURL('t3://testhost:7101')

foreignJNDIInstance.setUser('other\_server')

foreignJNDIInstance.setPassword('other\_user')

# now configuring foreign links

foreignLinkInstance=foreignJNDIInstance.createForeignJNDILink('MyTestLink')

foreignLinkInstance.setLocalJNDIName('jndi/local/testserver/TestServerConnectionFactory')

foreignLinkInstance.setRemoteJNDIName('jndi/testserver/TestServerConnectionFactory')

else:

print 'Foreign JNDI provider with the name myTestJNDIProvider already exists !'

activate()

exit()

* setup\_mail\_session

# ................ import ........................

from java.util import Properties

###################################################################

def createMailSession():

try:

print 'Create EMail session ...';

edit();

startEdit();

cd('/')

myTestMailMbean = cmo.createMailSession('TestNotificationEmail');

cd('/MailSessions/TestNotificationEmail');

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')],

ObjectName))

myTestMailMbean.setJNDIName('mail/TestNotificationEmail');

properties = java.util.Properties();

properties.put('mail.to','lector@wlsscriptbook.com');

properties.put('mail.from','author@wlsscriptbook.com');

properties.put('mail.transport.protocol','smtp');

properties.put('mail.smtp.host','mail.wlsscriptbook.com');

properties.put('mail.smtp.port','25');

properties.put('mail.smtp.user','username');

properties.put('mail.smtp.password','password');

myTestMailMbean.setProperties(properties);

save();

activate();

except:

print 'Exception while create EMail session !';

dumpStack();

exit();

# ================================================================

# Main Code Execution

# ================================================================

if \_\_name\_\_== "main":

print '###################################################################';

print '# Test create Mail session #';

print '###################################################################';

print '';

connect('weblogic', 'test1234', 't3://localhost:7001');

createMailSession()

disconnect();

* create\_WorkManager

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Create a WorkManager

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def createWorkManager(maxthreads, capacitycount):

try:

edit()

startEdit()

# change to the domain selftuning instance

cd('/SelfTuning/'+domainName)

# create a maxthreads contraint instance

cmo.createMaxThreadsConstraint('testMaxThreads')

cd('/SelfTuning/'+domainName+'/MaxThreadsConstraints/testMaxThreads')

# set count

cmo.setCount(maxthreads)

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')],

ObjectName))

# change to the domain selftuning instance

cd('/SelfTuning/'+domainName)

# create capacity instance

cmo.createCapacity('testCapacity')

cd('Capacities/testCapacity')

set('Count',capacitycount)

set('Notes','Defining a capacity instance')

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')],

ObjectName))

# now create a WorkManager

cd('/SelfTuning/'+domainName)

myWorkManager = cmo.createWorkManager('myTestWorkManager')

cd('/SelfTuning/'+domainName+'/WorkManagers/myTestWorkManager')

set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')],

ObjectName))

# set the maxthreads attribute to the created instance

cmo.setMaxThreadsConstraint(getMBean('/SelfTuning/'+domainName+

'/MaxThreadsConstraints/testMaxThreads'))

# set the capacity attribute to the created instance

cmo.setCapacity(getMBean('/SelfTuning/'+domainName+'/Capacities/testCapacity'))

save()

activate()

except Exception, e:

print 'Error in script:', e

cancelEdit()

* setup\_virtual\_host

# go to root

cd('/')

# create a new virtual host instance

cmo.createVirtualHost('Test\_VirtualHost\_1')

# go to the new virtual host

cd('/VirtualHosts/Test\_VirtualHost\_1')

# set the possible host names

set('VirtualHostNames',jarray.array([String('wlsautomation.com'), String('www.wlsautomation.com'), String('wlst\_and\_jmx.wlsautomation.com')], String))

# define which network channel this virtual should monitor for incoming requests

cmo.setNetworkAccessPoint('default')

# we can also define log settings for this virtual host in order to separate logs of the different servers

cd('/VirtualHosts/Test\_VirtualHost\_1/WebServerLog/Test\_VirtualHost\_1')

cmo.setRotationType('byTime')

cmo.setRotationTime('02:00')

cmo.setNumberOfFilesLimited(true)

cmo.setFileName('logs/virtualHosts\_1/Test\_VirtualHost\_1/access.log')

# go to the virtual host in order to define the targets

cd('/VirtualHosts/Test\_VirtualHost\_1')

# define the target(s) for this virtual host

set('Targets',jarray.array([ObjectName('com.bea:Name=AdminServer,Type=Server')], ObjectName))

* network\_channel\_and\_virtual\_hosts

# change to the managed-server where you want to create the services

cd('/Servers/Martin\_VirtualHostTest\_Domain\_MS1')

# create the first channel and allow only the http protocol

cmo.createNetworkAccessPoint('VH1Channel\_MS1')

cd('/Servers/Martin\_VirtualHostTest\_Domain\_MS1/NetworkAccessPoints/VH1Channel\_MS1')

cmo.setProtocol('http')

cmo.setListenPort(22012)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

# create the second channel and allow only the http protocol

cd('/Servers/Martin\_VirtualHostTest\_Domain\_MS1')

cmo.createNetworkAccessPoint('VH2Channel\_MS1')

cd('/Servers/Martin\_VirtualHostTest\_Domain\_MS1/NetworkAccessPoints/VH2Channel\_MS1')

cmo.setProtocol('http')

cmo.setListenPort(22013)

cmo.setEnabled(true)

cmo.setHttpEnabledForThisProtocol(true)

cmo.setTunnelingEnabled(false)

cmo.setOutboundEnabled(false)

cmo.setTwoWaySSLEnabled(false)

cmo.setClientCertificateEnforced(false)

# change to the root edit mbean as the virtual hosts have to be created on domain level !

cd('/')

# create the first virtual host

cmo.createVirtualHost('VirtualHost-1')

cd('/VirtualHosts/VirtualHost-1')

# link this virtual host to the first network channel

cmo.setNetworkAccessPoint('VH1Channel\_MS1')

set('VirtualHostNames',jarray.array([String('www.xyz.org')], String))

set('Targets',jarray.array([ObjectName('com.bea:Name=Martin\_VirtualHostTest\_Domain\_MS1,Type=Server')], ObjectName))

# create the second irtual host

cd('/')

cmo.createVirtualHost('VirtualHost-2')

cd('/VirtualHosts/VirtualHost-2')

# link it to the second channel

cmo.setNetworkAccessPoint('VH2Channel\_MS1')

set('VirtualHostNames',jarray.array([String('www.abc.org')], String))

set('Targets',jarray.array([ObjectName('com.bea:Name=Martin\_VirtualHostTest\_Domain\_MS1,Type=Server')], ObjectName))

* WTC\_configuration

def createAllWTCServer():

try:

totalWTCServer=domainProps.get("wtc.amountserver")

edit()

startEdit()

print 'Creating All WTC server ....'

i=1

while (i <= int(totalWTCServer)) :

try:

cd('/')

wtc\_name = get\_instance\_property('wtc.server',str(i), 'name');

wtc\_targetmanagedserver= get\_instance\_property('wtc.server',str(i), 'targetmanagedserver');

wtc\_localdomainname = get\_instance\_property('wtc.server',str(i), 'localtuxdomain.name');

wtc\_access\_point = get\_instance\_property('wtc.server',str(i),

'localtuxdomain.access\_point');

wtc\_access\_point\_id = get\_instance\_property('wtc.server',str(i),

'localtuxdomain.access\_point\_id');

wtc\_connection\_policy = get\_instance\_property('wtc.server',str(i),

'localtuxdomain.connection\_policy');

wtc\_nwaddr = get\_instance\_property('wtc.server',str(i), 'localtuxdomain.nw\_addr');

# create WTC server

cmo.createWTCServer(wtc\_name)

# change to WTC server

cd ('/WTCServers/'+wtc\_name)

cmo.addTarget(getMBean('/Servers/'+wtc\_targetmanagedserver))

# create local domain configuration

cmo.createWTCLocalTuxDom(wtc\_localdomainname)

cd ('WTCLocalTuxDoms/'+wtc\_localdomainname)

cmo.setAccessPoint(wtc\_access\_point)

cmo.setAccessPointId(wtc\_access\_point\_id)

cmo.setNWAddr(wtc\_nwaddr)

cmo.setConnectionPolicy(wtc\_connection\_policy)

# create remote tux domains

totalWTCRemoteDomains=get\_instance\_property('wtc.server',str(i), 'amountremotedomains')

r=1

while (r <= int(totalWTCRemoteDomains)) :

remotetuxdomain\_name = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.name');

remotetuxdomain\_access\_point = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.access\_point');

remotetuxdomain\_access\_point\_id = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.access\_point\_id');

remotetuxdomain\_local\_access\_point = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.local\_access\_point');

remotetuxdomain\_nw\_addr = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.nw\_addr');

remotetuxdomain\_federation\_url = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.federation\_url');

remotetuxdomain\_federation\_name = get\_instance\_property('wtc.server',str(i),

'remotetuxdomain.'+str(r)+'.federation\_name');

# create remote tux domain

cd ('/WTCServers/'+wtc\_name)

cmo.createWTCRemoteTuxDom(remotetuxdomain\_name)

cd ('WTCRemoteTuxDoms/'+remotetuxdomain\_name)

cmo.setAccessPoint(remotetuxdomain\_access\_point)

cmo.setAccessPointId(remotetuxdomain\_access\_point\_id)

cmo.setLocalAccessPoint(remotetuxdomain\_local\_access\_point)

cmo.setNWAddr(remotetuxdomain\_nw\_addr)

cmo.setFederationName(remotetuxdomain\_federation\_name)

cmo.setFederationURL(remotetuxdomain\_federation\_url)

r = r+1

# create WTC imports

totalWTCRImports=get\_instance\_property('wtc.server',str(i), 'amountimports')

r=1

while (r <= int(totalWTCRImports)) :

import\_name = get\_instance\_property('wtc.server',str(i), 'import.'+str(r)+'.name');

import\_resource\_name = get\_instance\_property('wtc.server',str(i),

'import.'+str(r)+'.resource\_name');

import\_remote\_name = get\_instance\_property('wtc.server',str(i),

'import.'+str(r)+'.remote\_name');

import\_local\_access\_point = get\_instance\_property('wtc.server',str(i),

'import.'+str(r)+'.local\_access\_point');

import\_remote\_access\_point = get\_instance\_property('wtc.server',str(i),

'import.'+str(r)+'.remote\_access\_point');

# create WTC import

cd ('/WTCServers/'+wtc\_name)

cmo.createWTCImport(import\_name)

cd ('WTCImports/'+import\_name)

cmo.setRemoteName(import\_remote\_name)

cmo.setLocalAccessPoint(import\_local\_access\_point)

cmo.setResourceName(import\_resource\_name)

cmo.setRemoteAccessPointList(import\_remote\_access\_point)

r = r+1

# create WTC exports

totalWTCRImports=get\_instance\_property('wtc.server',str(i), 'amountexports')

r=1

while (r <= int(totalWTCRImports)) :

export\_name = get\_instance\_property('wtc.server',str(i), 'export.'+str(r)+'.name');

export\_resource\_name = get\_instance\_property('wtc.server',str(i),

'export.'+str(r)+'.resource\_name');

export\_remote\_name = get\_instance\_property('wtc.server',str(i),

'export.'+str(r)+'.remote\_name');

export\_local\_access\_point = get\_instance\_property('wtc.server',str(i),

'export.'+str(r)+'.local\_access\_point');

export\_ejbname = get\_instance\_property('wtc.server',str(i),

'export.'+str(r)+'.ejbname');

# create WTC export

cd ('/WTCServers/'+wtc\_name)

cmo.createWTCExport(export\_name)

cd ('WTCExports/'+export\_name)

cmo.setRemoteName(export\_remote\_name)

cmo.setLocalAccessPoint(export\_local\_access\_point)

cmo.setResourceName(export\_resource\_name)

cmo.setEJBName(export\_ejbname)

r = r+1

print 'WTC Server: ',wtc\_name,', has been created Successfully !!!'

except:

dumpStack();

print '\*\*\*\*\* CANNOT CREATE WTC-Server with the Name : ' , wtc\_name ,' !'

print ''

i = i + 1

save()

activate()

except:

print 'Exception while creating WTC server !';

dumpStack();

* deletion\_example

import sys

# -------------- Check functions ------------------------------

# utility function to check weather a machine still has

# managed-server assigned to it

def machineHostsManagedServer(machineName):

cd ('/')

# get list of servers

listOfManagedServer = cmo.getServers()

# loop over the server mbeans and if the right one is found (or all if 'None' was provided) delete it

already\_found = false

for nextManagedServerInstance in listOfManagedServer:

# test if machine is defined and if yes if it is the machine in question

if (nextManagedServerInstance.getMachine()!=None):

if (nextManagedServerInstance.getMachine().getName() == machineName):

already\_found = true;

return already\_found

# utility function to check weather a cluster still has members

def clusterHasManagedServers(clusterName):

cd ('/')

# get the cluster mbean

clusterMBean = getMBean('/Clusters/'+clusterName)

# check if the number of servers in the cluster is > 0

if (len(clusterMBean.getServers()) > 0):

return true

else:

return false

# -------------- Delete functions ------------------------------

# delete a specific datasource or if 'None' is passed as argument delete all DS

def deleteDataSource(dataSourceName):

serverConfig()

# loop over the datasources mbeans and if the right one is found (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfDataSources = cmo.getJDBCSystemResources()

already\_found = 'false'

for datasourceInstance in listOfDataSources:

# if desired datasource is found OR no name (=None) was provided

if ((dataSourceName==None) or (dataSourceName == datasourceInstance.getName())):

# print name

print 'Datasource '+datasourceInstance.getName()+' will be destroyed !'

# first delete targets

datasourceInstance.setTargets(None)

# delete

cmo.destroyJDBCSystemResource(datasourceInstance)

# remember that DataSource was found

already\_found = 'true'

if ((dataSourceName!=None) and (already\_found=='false')):

print 'DataSource '+ dataSourceName + ' not found'

save()

activate()

serverConfig()

# delete a specific mail session or if 'None' is passed as argument delete all sessions

def deleteMailSessions(mailSessionName):

serverConfig()

# loop over the mail mbeans and if the right one is found (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfMailSessions = cmo.getJMailSessions()

already\_found = 'false'

for mailInstance in listOfMailSessions:

# if desired mail session is found OR no name (=None) was provided

if ((mailSessionName ==None) or (mailSessionName == mailInstance.getName())):

# print name

print Mail Session '+ mailInstance.getName()+' will be destroyed !'

# first delete targets

mailInstance.setTargets(None)

# delete

cmo.destroyMailSession(mailInstance)

# remember that Mail Session was found

already\_found = 'true'

if ((mailSessionName!=None) and (already\_found=='false')):

print Mail Session '+ mailSessionName + ' not found'

save()

activate()

serverConfig()

# delete a specific JNDI provider session or if 'None' is passed as argument delete all provider

def deleteForeignJNDIProvider(providerName):

serverConfig()

# loop over the foreign provider mbeans and if the right one is found (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfForeignProviders = cmo.getForeignJNDIProviders()

already\_found = 'false'

for foreignProviderInstance in listOfForeignProviders:

# if desired provider is found OR no name (=None) was provided

if ((providerName==None) or (providerName == foreignProviderInstance.getName())):

# print name

print 'Foreign provider '+foreignProviderInstance.getName()+' will be destroyed !'

# delete

cmo.destroyForeignJNDIProvider(foreignProviderInstance)

# remember that Provider was found

already\_found = 'true'

if ((providerName!=None) and (already\_found=='false')):

print 'Foreign provider '+ providerName + ' not found'

save()

activate()

serverConfig()

# delete a specific managed-server or if 'None' is passed as argument delete all managed-servers

# note that optionally the system can check if datasources, JMS providers or applications are still

# hosted on this server - then it will not delete it unless you pass true for the second option

def deleteManagedServer(managedServerName, deleteAlsoIfDependenciesExist=false):

serverConfig()

# loop over the server mbeans and if the right one is found (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfManagedServer = cmo.getServers()

already\_found = 'false'

for nextManagedServerInstance in listOfManagedServer:

# if desired MS is found OR no name (=None) was provided

if ((managedServerName==None) or (managedServerName == nextManagedServerInstance.getName())):

can\_be\_deleted = 'true';

if (str(deleteAlsoIfDependenciesExist)=='false'):

# check for dependencies

if (managedserverHostsApplications(managedServerName)):

can\_be\_deleted = 'false'

print 'Applications still deployed on '+

nextManagedServerInstance.getName()+' - cannot delete.'

if (managedserverHostsDatasources(managedServerName)):

can\_be\_deleted = 'false'

print nextManagedServerInstance.getName()+

' still hosts datasources - cannot delete.'

if (managedserverHostsJMSProviders(managedServerName)):

can\_be\_deleted = 'false'

print nextManagedServerInstance.getName()+

' still hosts JMS provider - cannot delete.'

if (can\_be\_deleted=='true'):

# print name

print 'Managed Server '+nextManagedServerInstance.getName()+' will be destroyed !'

# finally shutdown the server

print "Stopping " + nextManagedServerInstance.getName();

# shutdown the server if this server is not yet in state shutdown

# we need to switch to the domainRuntime in order to find out

domainRuntime()

serverRuntime = getMBean('/ServerLifeCycleRuntimes/'

+nextManagedServerInstance.getName())

serverState = serverRuntime.getState()

edit()

print 'Server '+ nextManagedServerInstance.getName()+' is in state ' + serverState

if serverState != 'SHUTDOWN':

try:

shutdown(nextManagedServerInstance.getName(),'Server','true'

,1000,force='true', block='true')

except:

pass

serverState = serverRuntime.getState()

print 'Server '+ nextManagedServerInstance.getName()+' is NOW in state '

+ serverState

# detach from cluster if any

nextManagedServerInstance.setCluster(None)

# detach from machine if any

nextManagedServerInstance.setMachine(None)

# finally delete

cmo.destroyServer(nextManagedServerInstance)

# remember that Provider was found

already\_found = 'true'

if ((managedServerName!=None) and (already\_found=='false')):

print 'Managed-Server '+ managedServerName + ' not found'

save()

activate()

serverConfig()

# delete a specific cluster or if 'None' is passed as argument delete all clusters

# note that optionally the system can check if managed-servers are still members of this cluster.

# It will not delete it unless you pass true for the second option. In the later case this

# function has to detach the server(s) from the cluster first, otherwise it cannot be deleted

def deleteCluster(clusterName, deleteAlsoIfDependenciesExist=false):

# loop over the cluster mbeans and if the right one is found

# (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfCluster = cmo.getClusters()

already\_found = 'false'

for nextClusterInstance in listOfCluster:

# if desired cluster is found OR no name (=None) was provided

if ((clusterName==None) or (clusterName == nextClusterInstance.getName())):

can\_be\_deleted = 'true';

if (str(deleteAlsoIfDependenciesExist)=='false'):

# check for dependencies

if (clusterHasManagedServers(nextClusterInstance.getName())):

can\_be\_deleted = 'false'

print 'Cluster '+nextClusterInstance.getName()+

' still has server members - cannot delete.'

if (can\_be\_deleted=='true'):

# print name

print 'Cluster '+nextClusterInstance.getName()+' will be destroyed !'

# if the cluster still has managed-servers, then detach them from the cluster first !

listOfManagedServer = nextClusterInstance.getServers()

for nextManagedServerInstance in listOfManagedServer:

# detach from cluster if any

nextManagedServerInstance.setCluster(None)

# finally delete the cluster

try:

cmo.destroyCluster(nextClusterInstance)

except Exception, ex:

print ex

# remember that cluster was found

already\_found = 'true'

if ((clusterName!=None) and (already\_found=='false')):

print 'Cluster '+ clusterName + ' not found'

save()

activate()

# delete a specific machine or if 'None' is passed as argument delete all machines

# note that optionally the system can check if managed-servers are still hosted on this machine.

# In this case it will not delete it unless you pass true for the second option

def deleteMachine(machineName, deleteAlsoIfDependenciesExist=false):

# loop over the machine mbeans and if the right one is found (or all if 'None' was provided) delete it

edit()

startEdit()

cd ('/')

listOfMachines = cmo.getMachines()

already\_found = 'false'

for nextMachineInstance in listOfMachines:

# if desired machine is found OR no name (=None) was provided

if ((machineName==None) or (machineName == nextMachineInstance.getName())):

can\_be\_deleted = 'true';

if (str(deleteAlsoIfDependenciesExist)=='false'):

# check for dependencies

if (machineHostsManagedServer(nextMachineInstance.getName())):

can\_be\_deleted = 'false'

print 'Machine '+nextMachineInstance.getName()+

' still has server members - cannot delete.'

if (can\_be\_deleted=='true'):

# print name

print 'Machine '+nextMachineInstance.getName()+' will be destroyed !'

# finally delete the machine

cmo.destroyMachine(nextMachineInstance)

# remember that machine was found

already\_found = true

if ((machineName!=None) and (already\_found=='false')):

print 'Machine '+ machineName + ' not found'

save()

activate()

# ================================================================

# Main Code Execution

# ================================================================

if \_\_name\_\_== "main":

print '#######################################################################';

print '# Weblogic resource deletion #';

print '#######################################################################';

print 'Usage: <user> <password> <URL> <resource-type> <None or resource name> <true/false for deleteIfDep>\n';

wls\_user = sys.argv[1]

wls\_password = sys.argv[2]

wls\_url = sys.argv[3]

wls\_typeForDeletion = sys.argv[4]

wls\_whatShouldBeDeleted = sys.argv[5] # can be 'None'

deleteAlsoIfDependenciesExist = sys.argv[6] # can be 'false' (should be) or 'true'

# connect

connect(wls\_user, wls\_password, wls\_url)

if (wls\_whatShouldBeDeleted != 'None'):

whatShouldBeDeleted = wls\_whatShouldBeDeleted

else:

whatShouldBeDeleted = None

if (wls\_typeForDeletion == 'Machine'):

deleteMachine(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

elif (wls\_typeForDeletion == 'Cluster'):

deleteCluster(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

elif (wls\_typeForDeletion == 'ManagedServer'):

deleteManagedServer(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

elif (wls\_typeForDeletion == 'DataSource'):

deleteDataSource(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

elif (wls\_typeForDeletion == 'Mail'):

deleteMailSessions(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

elif (wls\_typeForDeletion == 'JNDI'):

deleteForeignJNDIProvider(whatShouldBeDeleted,deleteAlsoIfDependenciesExist)

else:

print '\nUNKOWN resource type: '+wls\_typeForDeletion

* secret\_access\_files

import sys

adminURL = '';

adminUserName = '';

adminPassword = '';

userConfigFile = '';

userKeyFile = '';

# Load properties

def intialize():

global adminURL;

global adminUserName;

global adminPassword;

global userConfigFile;

global userKeyFile;

# test arguments

if len(sys.argv) != 6:

print 'Usage: wlst.sh createSecretAccess.py <admin-URL> <wls\_username> <wls\_password> <destUserConfigFile> <destUserKeyFile>';

exit();

try:

adminURL = sys.argv[1];

adminUserName = sys.argv[2];

adminPassword = sys.argv[3];

userConfigFile= sys.argv[4];

userKeyFile = sys.argv[5];

except:

print 'Cannot load arguments !';

exit();

print 'Initialization completed';

###################################################################

# Connect to adminserver

###################################################################

def connnectToAdminServer():

print 'Connecting to the Admin Server ';

connect(adminUserName, adminPassword, adminURL);

###################################################################

# Create userconfig and key

###################################################################

def creatSecretAccessFiles():

print 'Create secret access files !';

storeUserConfig(userConfigFile , userKeyFile);

###################################################################

# disconnect from adminserver

###################################################################

def disconnectFromAdminserver():

print 'Disconnect from the Admin Server...';

disconnect();

# ================================================================

# Main Code Execution

if \_\_name\_\_== "main":

print '';

intialize();

connnectToAdminServer();

creatSecretAccessFiles();

disconnectFromAdminserver();

* change\_PKI\_SSL\_settings

# connect to the admin server

connect('weblogic','Welcome1','t3://1.2.3.4:7001')

# go into edit mode

edit()

startEdit()

# change to the adinserver configuration

cd('/Servers/AdminServer')

# change keystore mode to custom and trust - means own keystore and truststore

set('KeyStores','CustomIdentityAndCustomTrust')

# set the keystore with our identity - our private key

set('CustomIdentityKeyStoreFileName','/myfiles/keystores/MartinIdentityKeyStore.jks')

set('CustomIdentityKeyStoreType','jks')

# passphrase for the keystore - remember that the private key is an important security token and must be protected

# this passthrase will be stored encrypted in WebLogic

set('CustomIdentityKeyStorePassPhrase','passwordForKeystore')

# configure the truststore with either the root CA's or individual public keys which can be trusted

set('CustomTrustKeyStoreFileName','/myfiles/keystores/MartinTrustKeyStore.jks')

set('CustomTrustKeyStoreType','jks')

set('CustomTrustKeyStorePassPhrase','passwordForKeystore')

# switch to the SSL section of the admin-server

cd('/Servers/AdminServer/SSL/AdminServer')

# Tell weblogic to use keystores for trust

set('IdentityAndTrustLocations','KeyStores')

# Tell the adminserver WHICH certificate of the identity keystore should be used as its server certificate

set('ServerPrivateKeyAlias','myCertificateAlias')

set('ServerPrivateKeyPassPhrase','MachinePassPhrase')

# disable hostname verifier. This should only be deactivated if you are on non-production systems and it

# might be possible that you have to accept certificates which do not belong to the partner's host

set('HostnameVerifier','None')

set('TwoWaySSLEnabled','true')

set('CertAuthenticator','')

# switching this to false means that the server does not force a client to present a client certificate.

# Should be ok for most (even production) environments

set('ClientCertificateEnforced','false')

save()

activate()

disconnect()

* secure\_servers

def setPKISSLonAllServers(domainProps):

print 'Iterate over all managed servers and set PKI / SSL configurations

domainConfig()

svrs = cmo.getServers()

edit()

startEdit()

for server in svrs:

# get server name

myServerName = server.getName()

# get server speecific parameter from property list

id\_keystore = domainProps.getProperty(myServerName+'.id\_keystore')

id\_keystorepassphrase = domainProps.getProperty(myServerName+'.id\_keystorepassphrase')

trust\_keystore = domainProps.getProperty(myServerName+'.trust\_keystore')

trust\_keystorepassphrase = domainProps.getProperty(myServerName+'.trust\_keystorepassphrase')

privatekeyalias = domainProps.getProperty(myServerName+'.privatekeyalias')

privatekeypassphrase = domainProps.getProperty(myServerName+'.privatekeypassphrase')

# change to server directory

cd('/Servers/'+myServerName)

# change keystore mode to custom and trust - means own keystore and truststore

cmo.setKeyStores('CustomIdentityAndCustomTrust')

# set the keystore with our identity - our private key

cmo.set('CustomIdentityKeyStoreFileName',id\_keystore)

cmo.set('CustomIdentityKeyStoreType','jks')

# passphrase for the keystore

# remember that the private key is an important security token and must be protected

# this passthrase will be stored encrypted in WebLogic

cmo.set('CustomIdentityKeyStorePassPhrase',id\_keystorepassphrase)

# configure the truststore with either the root CA's or individual public

cmo.set('CustomTrustKeyStoreFileName',trust\_keystore)

cmo.set('CustomTrustKeyStoreType','jks')

cmo.set('CustomTrustKeyStorePassPhrase',trust\_keystorepassphrase)

# switch to the SSL section of the admin-server

cd('/Servers/'+myServerName+'/SSL/'+myServerName)

# Tell weblogic to use keystores for trust

cmo.set('IdentityAndTrustLocations','KeyStores')

# Tell the server WHICH certificate of the identity keystore should be

cmo.set('ServerPrivateKeyAlias',privatekeyalias)

cmo.set('ServerPrivateKeyPassPhrase',privatekeypassphrase)

# disable hostname verifier

cmo.set('HostnameVerifier','None')

cmo.set('TwoWaySSLEnabled','true')

cmo.set('CertAuthenticator','')

# switching this to false means that the server does not force

# a client to present a client certificate.

# Should be ok for most (even production) environments

cmo.set('ClientCertificateEnforced','false')

save()

activate()

* class\_definition

from weblogic.management.security.authentication import UserReaderMBean

from weblogic.management.security.authentication import GroupReaderMBean

from weblogic.management.security.authentication import UserLockoutManagerMBean

from weblogic.management.security.authentication import UserLockoutManagerMBeanserverRuntime

class UserGroupManagement:

def \_\_init\_\_(self, realmName):

self.realmName = realmName

def \_\_init\_\_(self):

self.realmName = "myrealm"

#######################################################

# PART-1: List Information

#######################################################

# List all users in all authentication providers

def listAllUsers(self):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

allAuthenticationProviders = cmo.getAuthenticationProviders()

print 'List all Users in all Authentication Providers:'

for provider in allAuthenticationProviders:

if isinstance(provider,UserReaderMBean):

nextUserAuthenticationProvider = provider

print 'All users available in provider:'+provider.getName()+' in realm:'+self.realmName

cursor = provider.listUsers("\*",0)

while nextUserAuthenticationProvider.haveCurrent(cursor):

print ' User: ' + nextUserAuthenticationProvider.getCurrentName(cursor)

nextUserAuthenticationProvider.advance(cursor)

nextUserAuthenticationProvider.close(cursor)

except:

dumpStack()

print "Error in listAllUsers"

# return all user names in all authentication providers

def returnAllUserNames(self):

try:

userList = []

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

allAuthenticationProviders = cmo.getAuthenticationProviders()

for provider in allAuthenticationProviders:

if isinstance(provider,UserReaderMBean):

nextUserAuthenticationProvider = provider

cursor = provider.listUsers("\*",0)

while nextUserAuthenticationProvider.haveCurrent(cursor):

userList.append(nextUserAuthenticationProvider.getCurrentName(cursor))

nextUserAuthenticationProvider.advance(cursor)

nextUserAuthenticationProvider.close(cursor)

return userList

except:

dumpStack()

print "Error in returnAllUserNames"

# List all groups in all authentication providers

def listAllGroups(self):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

allAuthenticationProviders = cmo.getAuthenticationProviders()

print 'List all Groups in all Authentication Providers:'

for provider in allAuthenticationProviders:

if isinstance(provider,GroupReaderMBean):

nextGroupAuthenticationProvider = provider

print 'All groups available in provider:'+provider.getName()+' in realm:'+self.realmName

cursor = provider.listGroups("\*",0)

while nextGroupAuthenticationProvider.haveCurrent(cursor):

print ' Group: ' + nextGroupAuthenticationProvider.getCurrentName(cursor)

nextGroupAuthenticationProvider.advance(cursor)

nextGroupAuthenticationProvider.close(cursor)

except:

dumpStack()

print "Error in listAllGroups"

# return all group names in all authentication providers

def returnAllGroupNames(self):

try:

groupList = []

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

allAuthenticationProviders = cmo.getAuthenticationProviders()

for provider in allAuthenticationProviders:

if isinstance(provider,GroupReaderMBean):

nextGroupAuthenticationProvider = provider

cursor = provider.listGroups("\*",0)

while nextGroupAuthenticationProvider.haveCurrent(cursor):

groupList.append(nextGroupAuthenticationProvider.getCurrentName(cursor))

nextGroupAuthenticationProvider.advance(cursor)

nextGroupAuthenticationProvider.close(cursor)

return groupList

except:

dumpStack()

print "Error in returnAllGroupNames"

# List all user in the different groups in all authentication providers

def listUsersInGroups(self):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

allAuthenticationProviders = cmo.getAuthenticationProviders()

print 'List users in all groups in all Authentication Providers:'

for provider in allAuthenticationProviders:

if isinstance(provider,GroupReaderMBean):

nextGroupAuthenticationProvider = provider

print 'All user/groups available in provider:'+provider.getName()+

' in realm:'+self.realmName

cursor = nextGroupAuthenticationProvider.listGroups("\*",0)

while nextGroupAuthenticationProvider.haveCurrent(cursor):

nextGroup = nextGroupAuthenticationProvider.getCurrentName(cursor)

usersInActualGroup = provider.listAllUsersInGroup(nextGroup,"\*",0)

print ' Group: ' + nextGroupAuthenticationProvider.getCurrentName(cursor)

for nextUser in usersInActualGroup:

print ' User: '+nextUser

nextGroupAuthenticationProvider.advance(cursor)

nextGroupAuthenticationProvider.close(cursor)

except:

dumpStack()

print "Error in listUsersInGroups"

#######################################################

# PART-2: Create basic security artefacts

#######################################################

# create a new user in the default authenticator

def createUser(self,newUserName, newUserPassword, newUserDescription, deleteUserFirstIfExists):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+'/AuthenticationProviders/DefaultAuthenticator')

if (cmo.userExists(newUserName)):

if (deleteUserFirstIfExists):

print 'User '+newUserName+' already exists - removing old user first !'

cmo.removeUser(newUserName)

else:

# cannot create !!

print 'User '+newUserName+' already exists - CANNOT create !'

return

# create user

cmo.createUser(newUserName, newUserPassword, newUserDescription)

except:

dumpStack()

# create a new group in the default authenticator

def createGroup(self,newGroupName, newGroupDescription, deleteGroupFirstIfExists):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+'/AuthenticationProviders/DefaultAuthenticator')

if (cmo.groupExists(newGroupName)):

if (deleteGroupFirstIfExists):

print 'Group '+newGroupName+' already exists - removing old group first !'

cmo.removeGroup(newGroupName)

else:

# cannot create !!

print 'Group '+newGroupName+' already exists - CANNOT create !'

return

# create group

cmo.createGroup(newGroupName, newGroupDescription)

except:

dumpStack()

# add a user to a group. Group membership is very important for correct security rules

def addUserToGroup(self,userName, groupName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+'/AuthenticationProviders/DefaultAuthenticator')

# check if user exists

if (cmo.userExists(userName)==0):

print 'User '+userName+' does not exist CANNOT add '+userName+' to group '+groupName+' !'

return

# check if group exists

if (cmo.groupExists(groupName)==0):

print 'Group '+groupName+' does not exist CANNOT add '+userName+' to group '+groupName+' !'

return

# check if already member

if (cmo.isMember(groupName,userName,true)==1):

print 'User '+userName+' is already member of group '+groupName+' !'

return

# finally :-) add user to group

cmo.addMemberToGroup(groupName, userName)

except:

dumpStack()

# change the password of a user

def changeUserpassword(self,userName, oldPassword, newPassword):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+

'/AuthenticationProviders/DefaultAuthenticator')

# check if user exists

if (cmo.userExists(userName)==0):

print 'User '+userName+' does not exist - CANNOT change password !'

return

# change the password

cmo.changeUserPassword(userName, oldPassword, newPassword)

print "Changed password of user '+userName+' successfully"

except:

dumpStack()

# change the group memberships of this user

# this function will remove the user from all groups and add

# the user to all groups mentioned in the parameter

def changeGroupMembershipsOfUser(self, userName, allNewGroupNames):

try:

groupList = returnAllGroupNames()

# delete user from all groups where this user is member of

for groupName in groupList:

print 'test if '+userName+' is member of group '+groupName

if testIfUserIsMemberOfGroup(groupName, userName)==1:

# remove user

removeUserFromGroup(userName, groupName)

newGroupList = allNewGroupNames.split(',')

for groupName in newGroupList:

# add user to group

print 'Add '+userName+' to group '+groupName

addUserToGroup(userName, groupName)

except:

dumpStack()

#######################################################

# PART-3: Testing and information

#######################################################

# test if a user exists

def testIfUserExists(self, userName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+

'/AuthenticationProviders/DefaultAuthenticator')

return cmo.userExists(newUserName)

except:

dumpStack()

# test if a group exists

def testIfGroupExists(self, groupName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+

'/AuthenticationProviders/DefaultAuthenticator')

return cmo.groupExists(groupName)

except:

dumpStack()

# test if user is member of a group

def testIfUserIsMemberOfGroup(self, groupName, userName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+

'/AuthenticationProviders/DefaultAuthenticator')

return cmo.isMember(groupName,userName,true)

except:

dumpStack()

#######################################################

# PART-4: Delete user and group

#######################################################

# delete a user from the realm

def deleteUser(self, username):

# change to default authenticator

cd ('/SecurityConfiguration/' + domainName +

'/Realms/myrealm/AuthenticationProviders/DefaultAuthenticator')

# delete user

cmo.removeUser(userName)

# delete a group from the realm

def deleteGroup(self, groupname):

# change to default authenticator

cd ('/SecurityConfiguration/' + domainName +

'/Realms/myrealm/AuthenticationProviders/DefaultAuthenticator')

# delete group

cmo.removeGroup(groupname)

# remove user from group

def removeUserFromGroup(self, username, groupname):

# change to default authenticator

cd ('/SecurityConfiguration/' + domainName +

'/Realms/myrealm/AuthenticationProviders/DefaultAuthenticator')

remove user from group

if cmo.isMember(grouname,username,true):

cmo.removeMemberFromGroup(groupname,username)

#######################################################

# PART-5: Locking / Unlocking

#######################################################

# test if a user is lockedOut

def testIfUserAccountIsLocked(self, userName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

myLockoutManager = cmo.getUserLockoutManager()

return myLockoutManager.isLockedOut(userName)

except:

dumpStack()

# clear the lockedOut of a user and reactivate this user again

def clearUserAccountLock(self, userName):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

myLockoutManager = cmo.getUserLockoutManager()

myLockoutManager.clearLockout(userName)

print 'User account '+userName+' was unlocked !'

except:

dumpStack()

# list all user information

def listAllUserLockoutInformation(self):

try:

alluserNames = returnAllUserNames()

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName)

myLockoutManager = cmo.getUserLockoutManager()

print 'Lockout-Information about all user acoounts:'

for nextUser in alluserNames:

print ' User:'+nextUser+' isLocked:'+str(myLockoutManager.isLockedOut(nextUser))+

' LoginFailureCount:'+str(myLockoutManager.getLoginFailureCount(nextUser))+

' LastLoginFailure:'+str(myLockoutManager.getLastLoginFailure(nextUser))

except:

dumpStack()

# configure UserLockout Manager

# lockoutEnabled (boolean), lockoutThreshold (int), lockoutDuration (int)

def configureUserLockoutManager(self, lockoutEnabled, lockoutThreshold, lockoutDuration):

try:

edit()

startEdit()

cd ('/')

ulm=cmo.getSecurityConfiguration().getDefaultRealm().getUserLockoutManager()

# lockout activated or not ?

ulm.setLockoutEnabled(lockoutEnabled)

# lockout threshold - when gets an account locked

ulm.setLockoutThreshold(lockoutThreshold)

# amount (in minutes) how long an account is locked

ulm.setLockoutDuration(lockoutDuration)

save()

activate()

except:

dumpStack()

# list all user information

def listAllUserLockoutInformation(self):

try:

cd('/SecurityConfiguration/'+domainName+'/Realms/'+self.realmName+'/UserLockoutManager/UserLockoutManager')

print 'Actual lockout-information from the domain '+domainName+' :'

print ' LockoutEnabled : ', get('LockoutEnabled')

print ' InvalidLoginAttemptsTotalCount : ',cmo.getInvalidLoginAttemptsTotalCount()

print ' InvalidLoginUsersHighCount : ',cmo.getInvalidLoginUsersHighCount()

print ' LockedUsersCurrentCount : ',cmo.getLockedUsersCurrentCount()

print ' LockoutCacheSize : ',cmo.getLockoutCacheSize()

print ' LockoutDuration : ',cmo.getLockoutDuration()

print ' LockoutGCThreshold : ',cmo.getLockoutGCThreshold()

print ' LockoutResetDuration : ',cmo.getLockoutResetDuration()

print ' LockoutThreshold : ',cmo.getLockoutThreshold()

print ' LoginAttemptsWhileLockedTotalCount : '

,cmo.getLoginAttemptsWhileLockedTotalCount()

print ' UnlockedUsersTotalCount : ',cmo.getUnlockedUsersTotalCount()

print ' UserLockoutTotalCount : ',cmo.getUserLockoutTotalCount()

except:

dumpStack()

* check\_providers

my1stProvider = 'MyOwnCustomAuthenticator'

my2ndProvider = 'MyOwnCustomIdentityAsserter'

my1stProviderControlFlag = 'REQUISITE'

my2ndProviderControlFlag = 'SUFFICIENT'

# Connect to administration server

connect(username, password, url)

# Check if provider-1 provider exists

try:

cd('/SecurityConfiguration/' + domainName + '/Realms/myrealm/AuthenticationProviders/'

+ my1stProvider)

except:

print 'The Authentication Provider ' + my1stProvider + ' does not exist.'

exit()

# Check if provider-2 provider exists

try:

cd('/SecurityConfiguration/' + domainName + '/Realms/myrealm/AuthenticationProviders/'

+ my2ndProvider)

except:

print 'The Authentication Provider ' + my2ndProvider + ' does not exist.'

exit()

print 'Changing the control flags and do a reorder of the providers'

# The changes have to be done on the EDIT mbean

edit()

startEdit()

cd('/')

# get the necessary mbeans.

# A cd(..) is not sufficient as the mbean references will be needed for the reorder

realm = getMBean('/SecurityConfiguration/' + domainName + '/Realms/' + realmName)

prv1 = getMBean('/SecurityConfiguration/' + domainName + '/Realms/' + realmName

+ '/AuthenticationProviders/' + my1stProvider)

prv2 = getMBean('/SecurityConfiguration/' + domainName + '/Realms/' + realmName

+ '/AuthenticationProviders/' + my2ndProvider)

# Update 1st provider

prv1.setControlFlag(my1stProviderControlFlag)

# Update 2nd provider

prv2.setControlFlag(my2ndProviderControlFlag)

# Reorder providers

realm.setAuthenticationProviders(jarray.array([prv1,prv2], weblogic.management.security.authentication.AuthenticationProviderMBean))

# Activate changes

save()

activate(block='true')

print 'Realm has been changed and changes has been activated.'

exit()

* auditing

cd('/')

cmo.setConfigurationAuditType('audit')

# set domain logging configurations

cd('/Servers/AdminServer/Log/AdminServer')

cmo.setLoggerSeverity('Warning')

cmo.setDomainLogBroadcastSeverity('Warning')

cmo.setLogFileSeverity('Info')

cmo.setStdoutSeverity('Info')

cmo.setMemoryBufferSeverity('Debug')

# change to the realm

cd('/SecurityConfiguration/MartinTest\_Domain/Realms/myrealm')

# create an auditor instance

cmo.createAuditor('NewTestDefAuditor', 'weblogic.security.providers.audit.DefaultAuditor')

# change to the new auditor

cd('/SecurityConfiguration/MartinTest\_Domain/Realms/myrealm/Auditors/NewTestDefAuditor')

# enable information level audit event logging

cmo.setInformationAuditSeverityEnabled(true)

# enable warning level audit event logging

cmo.setWarningAuditSeverityEnabled(true)

# enable failure level audit event logging

cmo.setFailureAuditSeverityEnabled(true)

# enable error level audit event logging

cmo.setErrorAuditSeverityEnabled(true)

cmo.setSeverity('WARNING')

# set active handler entries. This example sets all handlers available just to demonstrate what is available

set('ActiveContextHandlerEntries',jarray.array([String('com.bea.contextelement.channel.Address'), String('com.bea.contextelement.channel.ChannelName'), String('com.bea.contextelement.channel.Port'), String('com.bea.contextelement.channel.Protocol'), String('com.bea.contextelement.channel.PublicAddress'), String('com.bea.contextelement.channel.PublicPort'), String('com.bea.contextelement.channel.RemoteAddress'), String('com.bea.contextelement.channel.RemotePort'), String('com.bea.contextelement.channel.Secure'), String('com.bea.contextelement.ejb20.Parameter'), String('com.bea.contextelement.entitlement.EAuxiliaryID'), String('com.bea.contextelement.jmx.AuditProtectedArgInfo'), String('com.bea.contextelement.jmx.ObjectName'), String('com.bea.contextelement.jmx.OldAttributeValue'), String('com.bea.contextelement.jmx.Parameters'), String('com.bea.contextelement.jmx.ShortName'), String('com.bea.contextelement.jmx.Signature'), String('com.bea.contextelement.saml.MessageSignerCertificate'), String('com.bea.contextelement.saml.SSLClientCertificateChain'), String('com.bea.contextelement.saml.subject.ConfirmationMethod'), String('com.bea.contextelement.saml.subject.dom.KeyInfo'), String('com.bea.contextelement.security.ChainPrevailidatedBySSL'), String('com.bea.contextelement.servlet.HttpServletRequest'), String('com.bea.contextelement.servlet.HttpServletResponse'), String('com.bea.contextelement.webservice.Integrity'), String('com.bea.contextelement.wli.Message'), String('com.bea.contextelement.wsee.SOAPMessage'), String('com.bea.contextelement.xml.SecurityToken'), String('com.bea.contextelement.xml.SecurityTokenAssertion')], String))

* setup\_security\_realm

edit()

startEdit()

cd('/SecurityConfiguration/MartinTestDomain')

cmo.createRealm('TestRealm')

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm')

cmo.setDeployCredentialMappingIgnored(false)

# Activate that change so that the realm is added to the runtime mbean server

save()

activate(block="true")

# now create the desired provider, mappers, ...

## setup default authenticator:

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm')

cmo.createAuthenticationProvider('DefaultAuthenticator',

'weblogic.security.providers.authentication.DefaultAuthenticator')

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm/AuthenticationProviders/DefaultAuthenticator')

cmo.setControlFlag('SUFFICIENT')

# identity asserter

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm')

cmo.createAuthenticationProvider('DefaultIdentityAsserter',

'weblogic.security.providers.authentication.DefaultIdentityAsserter')

cd('/SecurityConfiguration/' + domainName + '/Realms/'+ realmName +

'/AuthenticationProviders/DefaultIdentityAsserter')

set('ActiveTypes',jarray.array([String('AuthenticatedUser')], String))

# SAML identity asserter

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm')

cmo.createAuthenticationProvider('federation\_saml\_asserter',

'weblogic.security.providers.saml.SAMLIdentityAsserterV2')

# create other provider

cd('/SecurityConfiguration/MartinTestDomain/Realms/TestRealm')

# create the default authorization provider which is based on XACML

cmo.createAuthorizer('XACMLAuthorizer', 'weblogic.security.providers.xacml.authorization.XACMLAuthorizer')

# create the default password validator

cmo.createPasswordValidator('SystemPasswordValidator',

'com.bea.security.providers.authentication.passwordvalidator.SystemPasswordValidator')

# create the default adjudicator

cmo.createAdjudicator('DefaultAdjudicator',

'weblogic.security.providers.authorization.DefaultAdjudicator')

# create the default role mapping provider

cmo.createRoleMapper('DefaultRoleMapper',

'weblogic.security.providers.authorization.DefaultRoleMapper')

# create the default credential mapper

cmo.createCredentialMapper('DefaultCredentialMapper',

'weblogic.security.providers.credentials.DefaultCredentialMapper')

save()

activate(block="true")

* web\_service

connect('weblogic','Welcome1','t3://127.0.0.1:7071')

edit()

startEdit()

cd('/WebserviceSecurities')

create('default\_wss','WebserviceSecurities')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders')

create('Weblogic DK Credential Provider','WebserviceCredentialProviders')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider')

set('ClassName','weblogic.wsee.security.wssc.v200502.dk.DKCredentialProvider')

set('TokenType','dk')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('ConfidentialityKeyAlias','ConfigurationProperties')

cd('ConfidentialityKeyAlias')

set('Value','myAppAlias')

cd('../ConfidentialityKeyAlias')

set('EncryptValueRequired','false')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('ConfidentialityKeyPassword','ConfigurationProperties')

cd('ConfidentialityKeyPassword')

set('EncryptedValue','mypassword')

cd('../ConfidentialityKeyPassword')

set('EncryptValueRequired','true')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('ConfidentialityKeyStore','ConfigurationProperties')

cd('ConfidentialityKeyStore')

set('Value','/ MyIdentityKeyStore.jks')

cd('../ConfidentialityKeyStore')

set('EncryptValueRequired','false')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('ConfidentialityKeyStorePassword','ConfigurationProperties')

cd('ConfidentialityKeyStorePassword')

set('EncryptedValue','mypassword')

cd('../ConfidentialityKeyStorePassword')

set('EncryptValueRequired','true')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('IntegrityKeyAlias','ConfigurationProperties')

cd('IntegrityKeyAlias')

set('Value','myAppAlias')

cd('../IntegrityKeyAlias')

set('EncryptValueRequired','false')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('IntegrityKeyPassword','ConfigurationProperties')

cd('IntegrityKeyPassword')

set('EncryptedValue','mypassword')

cd('../IntegrityKeyPassword')

set('EncryptValueRequired','true')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('IntegrityKeyStore','ConfigurationProperties')

cd('IntegrityKeyStore')

set('Value','/MyIdentityKeyStore.jks')

cd('../IntegrityKeyStore')

set('EncryptValueRequired','false')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('IntegrityKeyStorePassword','ConfigurationProperties')

cd('IntegrityKeyStorePassword')

set('EncryptedValue','mypassword')

cd('../IntegrityKeyStorePassword')

set('EncryptValueRequired','true')

cd('/WebserviceSecurities/default\_wss/WebserviceCredentialProviders/Weblogic DK Credential Provider/ConfigurationProperties')

create('Length','ConfigurationProperties')

cd('Length')

set('Value','16')

cd('../Length')

set('EncryptValueRequired','false')

save()

activate()

* XACML\_policy

<?xml version="1.0" encoding="UTF-8"?>

<Policy

PolicyId="urn:bea:xacml:2.0:entitlement:role:WLSAutomationUsers:" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">

<Description>Usr(test\_1)|Grp(group\_wlsa)</Description>

<Target>

<Resources>

<Resource>

<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">WLSAutomationUsers</AttributeValue>

<ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"

DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>

</ResourceMatch>

</Resource>

</Resources>

<Actions>

<Action>

<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names:tc:xacml:2.0:actions:enableRole</AttributeValue>

<ActionAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>

</ActionMatch>

</Action>

</Actions>

</Target>

<Rule Effect="Permit" RuleId="primary-rule">

<Condition>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

<!-- users and groups -->

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">test\_1</AttributeValue>

<SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"

DataType="http://www.w3.org/2001/XMLSchema#string" SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>

</Apply>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">group\_wlsa</AttributeValue>

<SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:2.0:subject:group"

DataType="http://www.w3.org/2001/XMLSchema#string" SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>

</Apply>

</Apply>

</Condition>

</Rule>

<Rule Effect="Deny" RuleId="deny-rule"/>

</Policy>

* switch\_console

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

pathSeparator = '/';

domainProps = Properties();

userConfigFile = '';

userKeyFile = '';

consoleAction = '';

# Load properties

def intialize():

global domainProps;

global userConfigFile;

global userKeyFile;

global consoleAction;

# test arguments

if len(sys.argv) != 4:

print 'Usage: wlst.sh adminConsole.py <property\_file> ON/OFF';

exit();

try:

domainProps = Properties()

# load properties and overwrite defaults

input = FileInputStream(sys.argv[1])

domainProps.load(input)

input.close()

userConfigFile = sys.argv[2]

userKeyFile = sys.argv[3]

consoleAction = sys.argv[4]

except:

print 'Cannot load properties !'

exit();

print 'Initialization completed'

# Connect to adminserver

def connnectToAdminServer():

connUri = domainProps.getProperty('adminURI')

connect(userConfigFile=userConfigFile,userKeyFile=userKeyFile,url=connUri)

print 'Connected to '+connUri

# restart the admin server

def restartAdminServer():

try:

print 'Shutting down the Admin Server...';

shutdown(force='true', block='true')

print 'Start the admin server using start script';

domainLocation = domainProps.getProperty('domainsDirectory')

+ pathSeparator + domainProps.getProperty('domainName');

startScript = domainLocation+ pathSeparator + "startWebLogic.sh"

myCommand = 'nohup '+startScript+' > '+domainProps.getProperty('domainName')+'.log 2>&1 &'

print ('The following start command will be used: '+myCommand)

os.system(myCommand)

print 'Just in case wait for 10 seconds.'

java.lang.Thread.sleep(10000)

except OSError:

print 'Exception while rstarting the adminserver !'

dumpStack()

###################################################################

# set console visibility

###################################################################

def setConsoleVisibility():

try:

if ('ON' == consoleAction):

# mark it for edit

edit()

# start the edit

startEdit()

# set the flag to true

cmo.setConsoleEnabled(true)

# save the changes

save()

# activate the changes

activate()

# restart admin

restartAdminServer()

elif ('OFF' == consoleAction):

# mark it for edit

edit()

# start the edit

startEdit()

# set the flag to true

cmo.setConsoleEnabled(false)

# save the changes

save()

# activate the changes

activate()

# restart admin

restartAdminServer()

else:

print ('Operation '+ consoleAction+ ' is not supported !!')

except OSError:

print 'Exception while changing the console visibility !'

dumpStack();

# disconnect from adminserver

def disconnectFromAdminserver():

print 'Disconnect from the Admin Server...'

disconnect();

if \_\_name\_\_== "main":

print ' Toggle Admin Console usability '

print ''

intialize()

connnectToAdminServer()

setConsoleVisibility()

disconnectFromAdminserver()

* loop\_managed\_servers

# Start all managed servers of a domain

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

domainProps = Properties();

userConfigFile = '';

userKeyFile = '';

# Load properties

def intialize():

global domainProps;

global userConfigFile;

global userKeyFile;

# test arguments

if len(sys.argv) != 3:

print 'Usage: startAllManagedServers.py <property\_file>';

exit();

try:

domainProps = Properties()

# load properties and overwrite defaults

input = FileInputStream(sys.argv[1])

domainProps.load(input)

input.close()

userConfigFile = sys.argv[2]

userKeyFile = sys.argv[3]

except:

print 'Cannot load properties !';

exit();

print 'Initialization completed';

# Connect to adminserver - wait max. 10 minutes for connection

def connnectToAdminServer():

connUri = domainProps.getProperty('adminURL')

currentcount = 0;

adminServerIsRunning = 'false';

while ((adminServerIsRunning=='false') and (currentcount<30)):

try:

print 'Connecting to the Admin Server ('+connUri+')';

connect(userConfigFile=userConfigFile,userKeyFile=userKeyFile,url=connUri);

print 'Connected';

adminServerIsRunning = 'true';

except:

print 'AdminServer is (not yet) running. Will wait for 10sec.';

java.lang.Thread.sleep(10000);

currentcount = currentcount +1;

if (adminServerIsRunning=='false'):

print 'Could not connect to admin server - script will be exit !'

exit();

# get Server status

def getMSserverStatus(server):

try:

cd('/ServerLifeCycleRuntimes/' +server)

except:

print 'oh ohohohoh';

dumpStack();

return cmo.getState()

# Start all managed server

def startAllManagedServers():

try:

print 'Loop through the managed servers and start all servers ';

domainConfig()

svrs = cmo.getServers()

domainRuntime()

for server in svrs:

# Do not start the adminserver, it's already running

if server.getName() != 'AdminServer':

# Get state and machine

serverState = getMSserverStatus(server.getName())

print server.getName() + " is " + serverState

# startup if needed

if (serverState == "SHUTDOWN") or

(serverState == "FAILED\_NOT\_RESTARTABLE"):

print "Starting " + server.getName();

start(server.getName(),'Server')

serverState = getMSserverStatus(server.getName())

print "Now " + server.getName() + " is " + serverState;

except:

print 'Exception while starting managed servers !';

dumpStack();

# disconnect from adminserver

def disconnectFromAdminserver():

print 'Disconnect from the Admin Server...';

disconnect();

# ================================================================

# Main Code Execution

# ================================================================

if \_\_name\_\_== "main":

print ' Start all managed server of the domain \n';

intialize();

connnectToAdminServer();

startAllManagedServers();

disconnectFromAdminserver();

(Note that line breaks have been introduced for better readability)

* JDBCDataSourcRuntimeMBeans

wls:/MartinTest\_Domain/serverRuntime/JDBCServiceRuntime/MartinTest\_Domain\_MS3/JDBCDataSourceRuntimeMBeans/MyDS> ls()

dr-- JDBCDriverRuntime

dr-- LastTask

dr-- WorkManagerRuntimes

-r-- ActiveConnectionsAverageCount 0

-r-- ActiveConnectionsCurrentCount 0

-r-- ActiveConnectionsHighCount 1

-r-- ConnectionDelayTime 299

-r-- ConnectionsTotalCount 4

-r-- CurrCapacity 1

-r-- CurrCapacityHighCount 1

-r-- DatabaseProductName Oracle

-r-- DatabaseProductVersion Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production

With the Partitioning, OLAP, Data Mining and Real Application Testing options

-r-- DeploymentState 2

-r-- DriverName Oracle JDBC driver

-r-- DriverVersion 11.2.0.2.0

-r-- Enabled true

-r-- FailedReserveRequestCount 0

-r-- FailuresToReconnectCount 0

-r-- HighestNumAvailable 1

-r-- HighestNumUnavailable 1

-r-- LeakedConnectionCount 0

-r-- ModuleId MyPersistentBackend

-r-- Name MyPersistentBackend

-r-- NumAvailable 1

-r-- NumUnavailable 0

-r-- PrepStmtCacheAccessCount 0

-r-- PrepStmtCacheAddCount 0

-r-- PrepStmtCacheCurrentSize 0

-r-- PrepStmtCacheDeleteCount 0

-r-- PrepStmtCacheHitCount 0

-r-- PrepStmtCacheMissCount 0

-r-- Properties {oracle.net.wallet\_location=/opt/wallets/martintest}

-r-- ReserveRequestCount 6

-r-- State Running

-r-- Type JDBCDataSourceRuntime

-r-- VersionJDBCDriver oracle.jdbc.OracleDriver

-r-- WaitSecondsHighCount 0

-r-- WaitingForConnectionCurrentCount 0

-r-- WaitingForConnectionFailureTotal 0

-r-- WaitingForConnectionHighCount 0

-r-- WaitingForConnectionSuccessTotal 0

-r-- WaitingForConnectionTotal 0

-r-- WorkManagerRuntimes null

-r-x clearStatementCache Void :

-r-x dumpPool Void :

-r-x dumpPoolProfile Void :

-r-x forceShutdown Void :

-r-x forceSuspend Void :

-r-x poolExists Boolean : String(name)

-r-x preDeregister Void :

-r-x reset Void :

-r-x resume Void :

-r-x shrink Void :

-r-x shutdown Void :

-r-x start Void :

-r-x suspend Void :

-r-x testPool String :

* datatsource\_comprehensive\_example

# -------------------------------

# CLASS for database operations

# -------------------------------

class DatasourceOperations(object):

# Contructor which can take a datasourcename. None means all datasources

def \_\_init\_\_(self, datasourcename=None):

self.datasourcename = datasourcename

# ------------------------ Testing -----------------------------------------

# internal method (for testing one datasource on one specific serverruntime

def testDatasourceOnDestinationServer(self, destinationServerRuntime):

# get all datasource runtime mbeans

allDSRuntimesOnThisServer = destinationServerRuntime.getJDBCServiceRuntime().

getJDBCDataSourceRuntimeMBeans()

# loop over the runtime mbeans and if the right datasource is found test it

already\_found = false

for datasources in allDSRuntimesOnThisServer:

# if desired datasource is found OR no name (=None) was provided

if ((self.datasourcename==None) or (self.datasourcename == datasources.getName())):

# print name

print 'Datasource name is: '+datasources.getName()

# print state

print 'State is ' +datasources.getState()

# test datasource pool

print datasources.testPool()

# remember that DS was found

already\_found = true

if ((self.datasourcename!=None) and (already\_found==false)):

print 'Datasource '+ self.datasourcename + ' not found on server '

+destinationServerRuntime.getName()

# test one specific or all (if passed None as argument) datasources of the connected server

# note that the argment was passed to the constructor !

def testDatasourceOnConnectedServer(self):

# switch to server runtime

serverRuntime()

cd ('/')

# call the test method with the actual (cmo) server runtime

self.testDatasourceOnDestinationServer(cmo)

# test one specific or all (if passed None as argument) datasources of the connected server

# note that the argment was passed to the constructor !

def testDatasourceOnAllServersInDomain(self):

if (str(isAdminServer)=='true'):

domainConfig()

allServers = cmo.getServers()

domainRuntime()

for server in allServers:

print 'Test Datasources on server: '+server.getName()

cd ('/ServerRuntimes/'+server.getName())

# call testDatasourceOnDestinationServer

self.testDatasourceOnDestinationServer(cmo)

else:

print 'testDatasourceOnAllServersInDomain is only available '+

'if connected to the adminserver !'

return

# ------------------------ Lifecycle -----------------------------------------

# internal method for lifecycle operations of one datasource on one specific serverruntime

# lifecycleoperation can be 'start' or 'shutdown' or 'suspend' or 'resume'

def lifecycleDatasourceOnDestinationServer(self, destinationServerRuntime, lifecycleoperation):

# get all datasource runtime mbeans

allDSRuntimesOnThisServer = destinationServerRuntime.getJDBCServiceRuntime().

getJDBCDataSourceRuntimeMBeans()

# loop over the runtime mbeans and if the right datasource is found start it

already\_found = false

for datasources in allDSRuntimesOnThisServer:

# if desired datasource is found OR no name (=None) was provided

if ((self.datasourcename==None) or (self.datasourcename == datasources.getName())):

# print name

print 'Datasource name is: '+datasources.getName()

# print state

actualState = datasources.getState()

print 'State before operation is: ' + actualState

if ('start' == lifecycleoperation):

# start the server if the server is not already running

if (actualState != 'Running'):

datasources.start()

else:

print 'Datasource '+datasources.getName()+' is already running !'

elif ('shutdown' == lifecycleoperation):

# stop the server if the server is not already shutdown

if (actualState != 'Shutdown'):

datasources.shutdown()

else:

print 'Datasource '+datasources.getName()+' is already stopped !'

elif ('resume' == lifecycleoperation):

# resume the server if the server is not already running

if (actualState != 'Running'):

datasources.resume()

else:

print 'Datasource '+datasources.getName()+' is already running !'

elif ('suspend' == lifecycleoperation):

# suspend the server if the server is not already suspended

if (actualState != 'Suspended'):

datasources.suspend()

else:

print 'Datasource '+datasources.getName()+' is already suspended !'

else:

print '!!! Lifecycleoperation '+lifecycleoperation+' is not supported !'

# print state

print 'Datasource state is now: ' +datasources.getState()

already\_found = true

if ((self.datasourcename!=None) and (already\_found==false)):

print 'Datasource '+ self.datasourcename + ' not found on server '

+destinationServerRuntime.getName()

# lifecycle operations on the connected server only

def lifecycleDatasourceOnConnectedServer(self, lifecycleoperation):

# switch to server runtime

serverRuntime()

cd ('/')

self.lifecycleDatasourceOnDestinationServer(cmo, lifecycleoperation)

# lifecycle operations on all servers of the domain

def lifecycleDatasourceOnAllServersInDomain(self, lifecycleoperation):

if (str(isAdminServer)=='true'):

domainConfig()

allServers = cmo.getServers()

domainRuntime()

for server in allServers:

print lifecycleoperation+' all datasources on server: '+server.getName()

try:

cd ('/ServerRuntimes/'+server.getName())

# call testDatasourceOnDestinationServer

self.lifecycleDatasourceOnDestinationServer(cmo, lifecycleoperation)

except:

print 'Problem with server '+server.getName()

else:

print 'lifecycleoperationDatasourceOnAllServersInDomain('+lifecycleoperation+

') is only available if connected to the adminserver !'

return

# ------------------------ SHORTCUT methods -----------------------------------------

def startDatasourceOnAllServersInDomain(self):

self.lifecycleDatasourceOnAllServersInDomain('start')

def stopDatasourceOnAllServersInDomain(self):

self.lifecycleDatasourceOnAllServersInDomain('shutdown')

def suspendDatasourceOnAllServersInDomain(self):

self.lifecycleDatasourceOnAllServersInDomain('suspend')

def resumeDatasourceOnAllServersInDomain(self):

self.lifecycleDatasourceOnAllServersInDomain('resume')

def startDatasourceOnConnectedServer(self):

self.lifecycleDatasourceOnConnectedServer('start')

def stopDatasourceOnConnectedServer(self):

self.lifecycleDatasourceOnConnectedServer('shutdown')

def suspendDatasourceOnConnectedServer(self):

self.lifecycleDatasourceOnConnectedServer('suspend')

def resumeDatasourceOnConnectedServer(self):

self.lifecycleDatasourceOnConnectedServer('resume')

# ------------------------ Target / Untarget methods ---------------------------------

# untarget datasource. Note if "None" was set in the constructor, then all DS will untarget

def untargetDatasource(self):

if (str(isAdminServer)=='true'):

serverConfig()

# edit mode necessary

edit()

startEdit()

# get all datasource config mbeans

allDSconfigs = cmo.getJDBCSystemResources()

# loop over all datasources

already\_found = false

for datasources in allDSconfigs:

# if desired datasource is found OR no name (=None) was provided

if ((self.datasourcename==None) or (self.datasourcename == datasources.getName())):

# print name

print 'Datasource name is: '+datasources.getName()

# untarget

cd('/JDBCSystemResources/'+datasources.getName())

set('Targets',jarray.array([], ObjectName))

print ' Datasource has been untargeted !'

# remember that DS was found

already\_found = true

if ((self.datasourcename!=None) and (already\_found==false)):

print 'Datasource '+ self.datasourcename + ' not found in this domain !'

# activate changes

save()

activate()

else:

print 'untargetDatasource() is only available if connected to the adminserver !'

return

# target datasource.

# Note if "None" was set in the constructor, then all DS will target to the same target

# target type can be 'Server' or 'Cluster'

def targetDatasource(self, newTarget, targetType='Cluster'):

if (str(isAdminServer)=='true'):

serverConfig()

# edit mode necessary

edit()

startEdit()

# get all datasource config mbeans

allDSconfigs = cmo.getJDBCSystemResources()

# loop over all datasources

already\_found = false

for datasources in allDSconfigs:

# if desired datasource is found OR no name (=None) was provided

if ((self.datasourcename==None) or (self.datasourcename == datasources.getName())):

# print name

print 'Datasource name is: '+datasources.getName()

# target

cd('/JDBCSystemResources/'+datasources.getName())

set('Targets',jarray.array([ObjectName('com.bea:Name='+newTarget+',Type='+targetType)], ObjectName))

print ' Datasource has been targeted to '+targetType+' with the name '+newTarget+' !'

# remember that DS was found

already\_found = true

if ((self.datasourcename!=None) and (already\_found==false)):

print 'Datasource '+ self.datasourcename + ' not found in this domain !'

# activate changes

save()

activate()

else:

print 'untargetDatasource() is only available if connected to the adminserver !'

return

* WTCRuntime

# connect to the server

connect(...)

# switch to the domain ruuntime

domainRuntime()

# switch to the WTC runtime of a managed-server

cd ('/ServerRuntimes/MS1/WTCRuntime/WTCService')

# list

ls()

-r-- Name WTCService

-r-- Type WTCRuntime

-r-x getServiceStatus Integer : String(java.lang.String),String(java.lang.String),String(java.lang.String)

-r-x getServiceStatus Integer : String(localAccessPoint),String(svcName)

-r-x getServiceStatus Integer : String(localAccessPoint),String(svcName),Boolean(isImport)

-r-x getServiceStatus Integer : String(svcName)

-r-x getServiceStatus Integer : String(svcName),Boolean(isImport)

-r-x listConnectionsConfigured weblogic.wtc.gwt.DSessConnInfo[] :

-r-x preDeregister Void :

-r-x resumeService Void : String(localAccessPoint),String(remoteAccessPointList),String(svcName)

-r-x resumeService Void : String(localAccessPoint),String(svcName)

-r-x resumeService Void : String(localAccessPoint),String(svcName),Boolean(isImport)

-r-x resumeService Void : String(svcName)

-r-x resumeService Void : String(svcName),Boolean(isImport)

-r-x startConnection Void : String(LDomAccessPointId)

-r-x startConnection Void : String(LDomAccessPointId),String(RDomAccessPointId)

-r-x stopConnection Void : String(LDomAccessPointId)

-r-x stopConnection Void : String(LDomAccessPointId),String(RDomAccessPointId)

-r-x suspendService Void : String(localAccessPoint),String(remoteAccessPointList),String(svcName)

-r-x suspendService Void : String(localAccessPoint),String(svcName)

-r-x suspendService Void : String(localAccessPoint),String(svcName),Boolean(isImport)

-r-x suspendService Void : String(svcName)

-r-x suspendService Void : String(svcName),Boolean(isImport)

* deploy\_applications

import sys

#=======================================================

# Function for fresh plain deployment

#=======================================================

def newDeploy(appName,ver):

print 'Deploying .........'

deploy(appName,'/path/applications/'+appName+'/'+ver , targets='AdminServer')

startApplication(appName)

#=======================================================

# Function for finding the Application Status

#=======================================================

def appstatus(appName, serverName):

cd('domainRuntime:/AppRuntimeStateRuntime/AppRuntimeStateRuntime')

#get current real state for app in specific server

currentState = cmo.getCurrentState(appName, serverName)

return currentState

#======================================================

# Undeploy the given application

# Target we can change according to domain and application deployed on

#======================================================

def unDeploy(appName):

print 'stopping and undeploying ....'

stopApplication(appName, targets='AdminServer')

undeploy(appName, targets='AdminServer')

#========================================================

# Main program here...

# Target you can change as per your need

#========================================================

appName = sys.argv[1]

ver = sys.argv[2]

user = sys.argv[3]

passwd = sys.argv[4]

adminurl= sys.argv[5]

connect(user, passwd, adminurl)

cd('AppDeployments')

appflag=0

y=ls(returnMap='true')

for i in y :

if i.startswith(appName ) ==1:

#Checking for the application existence)

print i

print appstatus(i,'AdminServer')

if appstatus(i,'AdminServer')=='STATE\_RETIRED' :

appflag=1

break

elif appstatus(i,'AdminServer')=='STATE\_ACTIVE':

appflag=2

else:

print ' other Applications are Running '

pass

if appflag == 1 :

print 'application having RETIERED STATE ..'

unDeploy(i)

print appstatus(i,'AdminServer')

newDeploy(appName,ver)

print appstatus(i,'AdminServer')

elif appflag== 2:

print 'Application exists in ACTIVE state...'

newDeploy(appName,ver)

print appstatus(i,'AdminServer')

else:

print 'new application'

* check\_module\_targeting

import jarray

def retargetApplicationComponent(appName, componentName, newTargetListNames, newTargetListTypes):

# switch to domain runtime

domainRuntime()

# go to app runtime state

cd ('/AppRuntimeStateRuntime/AppRuntimeStateRuntime')

# check if appName is in the list of known applications

appIDs = cmo.getApplicationIds() # ApplicationIds - java.lang.String[vhtest]

if not (appName in appIDs):

print 'PROBLEM: Application ' + appName + ' is unknown !'

return

# get the modules of application

myModules = cmo.getModuleIds(appName)

# check if module exists in application

if not (componentName in myModules):

print 'PROBLEM: Module '+componentName+' does not exist in application ' + appName + ' !'

return

# ok application and component exist - therefore go ahead

# test if exactly all targets are already configured

allDone = true

myCurrentTargets = cmo.getModuleTargets('vhtest','VH1')

print 'Actual Targets: ' , myCurrentTargets

for index, value in enumerate(newTargetListNames):

print ' ... checking ' , value

if not (value in myCurrentTargets):

allDone = false

# check if current target list is longer than new target list

if len(myCurrentTargets) > len(newTargetListNames):

allDone = false

if allDone:

print 'The module '+componentName+' is already targeted to the desired list of targets - no retarget necessary !'

return

# ok - this means we need to retarget :-(

# start edit session

edit()

startEdit()

# go to app deployments configuration

cd ('/AppDeployments/vhtest')

# create subdeployment for module and ignore exception if already exist

try:

cmo.createSubDeployment('VH1')

except:

print 'Ignore submodule creation exception: please check if submodule already exists'

# go to the subdeployment

cd('SubDeployments/VH1')

# create new array with target ObjectNames

targetsForDeployment = []

# convert the argument lists into a Obe´jectName array

for index, value in enumerate(newTargetListNames):

nextName =str('com.bea:Name='+value+',Type='+newTargetListTypes[index])

targetsForDeployment.append(ObjectName(nextName))

# finally set the targets

set('Targets',jarray.array(targetsForDeployment, ObjectName))

# save changes and activate

save()

activate()

# back to domain runtime

domainRuntime()

print 'Retarget of '+componentName+' is done !'

if \_\_name\_\_== "main":

# connect to the server

connect('weblogic','<pw>','t3://localhost:7100')

# retarget

retargetApplicationComponent('vhtest', 'VH1', ['MyVirtualHost\_MS1','MyVirtualHost\_MS2'],

['VirtualHost','VirtualHost'])

print '\nScript has finished successfully '

* extended\_deployment

import os

import time

import sys

from java.util import Date

from java.text import SimpleDateFormat

# define global variables and constants

t = Date()

today=SimpleDateFormat("dd\_MMM\_HH:mm").format(Date())

user=sys.argv[1]

URL=sys.argv[2]

src="/home/"+user+"/CODE/"

bksrc="/home/"+user+"/CODE\_"

dpath="/home/domains/wd"+user

ECODE='\033[0m \n'

G='\033[1;40;32m'

R='\033[1;40;31m'

deploylist=['a\_ejb.jar','a\_adapter.jar','b\_adapter.jar','c.war']

# define the modular functions

def backup():

try:

#Creating the Backup with Date And Time

command = "cp -R "+src+" "+bksrc+today

os.system(command)

except:

print R+' Code Backup FAILED!! '+ECODE

print dumpStack()

def getCode():

try:

# GETTING THE Fresg Code from build location

os.system('scp user@hostname:/home/user/Code\_Ioc/\*.\*ar '+src)

print G+' THE CODE COPIED SUCCESSFULLY '+ECODE

except:

print R+' THE CODE COPYING FAILED?!?!?'+ECODE

print dumpStack()

exit()

def conn():

try:

# CONNECTING TO THE SERVER ....

connect(userConfigFile=UCF, userKeyFile=UKF, url=URL)

except:

print R+' CONNECTION FAILED....',+ECODE

print dumpStack()

exit()

def editing():

edit()

startEdit()

def activating():

save()

activate()

def stoppingApp():

deploylist.reverse()

for s in deploylist:

try:

editing()

progress=stopApplication(s,timeout=360000)

progress.printStatus()

activating()

except:

print R+' FAILED TO STOP THE APPLICATION '+ECODE

print dumpStack()

print G+' APPLICATION STOPPED '+ECODE

def pUndeploy():

deploylist.reverse()

for s in deploylist:

try:

editing()

progress=undeploy(s, timeout=360000)

progress.printStatus()

activating()

except:

print R+' FAILED TO UNDEPLOY THE APPLICATION '+ECODE

print dumpStack()

print G+' APPLICATION UNDEPLOYED '+ECODE

# This module is optional

def clearCache():

try:

print G+' CLEARING THE CACHE '+ECODE

command = "rm -rf "+dpath+"/servers/"+user+"/tmp/\*.\*"

os.system(command)

print G+' CLEARED THE CACHE '+ECODE

except:

print R+' FAILED TO CLEAR CACHE '+ECODE

def pDeploy():

for s in deploylist:

try:

editing()

progress=deploy(s,src+s,target=user,timeout=360000)

progress.printStatus()

activating()

except:

print R+'FAILED TO DEPLOY THE APPLICATION'+ECODE

print dumpStack()

print G+' APPLICATION DEPLOYED '+ECODE

def startingApp():

for s in deploylist:

try:

editing()

startApplication(s,timeout=360000)

activating()

except:

print R+'FAILED TO START THE APPLICATION '+ECODE

print G+'APPLICATION STARTED SUCCESSFULLY '+ECODE

if \_\_name\_\_== "main":

backup()

getCode()

conn()

stoppingApp()

pUndeploy()

clearCache()

pDeploy()

startingApp()

print G+' ....DEPLOYMENET DONE...'+ECODE

* queried\_values\_summary

def printServerSummary(outputprefix, servername, rootDirInTree):

cd(rootDirInTree)

print outputprefix+"Server SUMMARY Information:"

# print Name

print outputprefix+" Name ....... = " + get("Name")

print outputprefix+" State ...... = " + get("State");

print outputprefix+" HealthState = " + getHealthStateInformation(get("HealthState"));

cd (rootDirInTree+"JVMRuntime/"+servername)

print outputprefix+" HeapFreeCurrent ................. = " + str(get("HeapFreeCurrent"))

cd (rootDirInTree+"JTARuntime/JTARuntime")

# print HealthState

print outputprefix+" JTA - HealthState = " +

getHealthStateInformation(get("HealthState"));

# change to ThreadPoolRuntime

cd (rootDirInTree+"ThreadPoolRuntime/ThreadPoolRuntime")

print outputprefix+" ThreadPoolRuntime-HealthState ... = " +

getHealthStateInformation(get("HealthState"));

# print CompletedRequestCount

print outputprefix+" ThreadPoolRuntime-CompletedRequestCount = " +

str(get("CompletedRequestCount"))

cd(rootDirInTree)

print outputprefix+" Datasources:";

dataSources = cmo.getJDBCServiceRuntime().getJDBCDataSourceRuntimeMBeans()

if (len(dataSources) > 0):

for dataSource in dataSources:

print outputprefix+" Datasource-Name..... = "+ dataSource.getName()

print outputprefix+" Datasource-State.... = "+ dataSource.getState()

print("\n\n")

* server\_start\_information

import os

def ServerState():

servers = getRunningServerNames()

domainRuntime()

currentState=""

app=""

# iterate over all servers

for server in servers:

try:

cd("/ServerRuntimes/" + server.getName())

ActivationTime = cmo.getActivationTime()

strout=SimpleDateFormat('d MMM yyyy HH:mm:ss').format(java.util.Date(cmo.getActivationTime()))

except:

print 'Skipping ..'

print 'ServerNAME: %4s %10s' %(server.getName(),strout)

exit ()

# get the names of all servers in this domain

def getRunningServerNames():

domainConfig()

return cmo.getServers()

if \_\_name\_\_== "main":

connect (str(user),str(password),str(aserver))

redirect('Server.log', 'false')

ServerState()

* query\_server\_states

import os

def sendMail(domain,count,dest,sname):

str3 = 'Domain:#' + domain +'# Server Name: '+ sname +', '+ dest +' @' + str(count)

cmd = "echo " + str3 + " > /tmp/rw\_serverstate\_file"

os.system(cmd)

os.system('/bin/mailx -s "Caution, The server state is abnormal !!! " operators@test.com < /tmp/rw\_serverstate\_file')

print '\*\*\*\*\*\*\*\*\* ALTERT MAIL HAS BEEN SENT \*\*\*\*\*\*\*\*\*\*\*'

cmd = "rm /tmp/rw\_serverstate\_file"

os.system(cmd)

def ServerState():

servers = getRunningServerNames()

domainRuntime()

sign="###"

activecount=0

problemcount=0

currentState=""

app=""

for server in servers:

try:

cd("/ServerLifeCycleRuntimes/" + server.getName())

serverState = cmo.getState()

if serverState == "ADMIN" :

resume(server.getName(),block="true")

sendMail(aserver,app,serverState,server.getName())

problemcount=problemcount +1

elif serverState == "FAILED" or serverState == "UNKNOWN" or serverState == "STARTING":

sendMail(aserver,app,serverState,server.getName())

problemcount=problemcount +1

else:

activecount=activecount +1

except:

print 'Skipping ..'

print 'ServerNAME: %4s %10s' %(server.getName(),serverState)

print 'Total %4s RUNNING :\033[1;31m %4s Problem \033[0m' %(repr(activecount),repr(problemcount))

exit ()

def getRunningServerNames():

domainConfig()

return cmo.getServers()

if \_\_name\_\_== "main":

redirect('Server.log', 'false')

connect (str(user),str(password),str(adminserverURL))

ServerState()

(Script written by Atilla Demirel)

* get\_memory\_information

######################################################################

# Get memory information from all servers in your domain #

######################################################################

# test arguments

if len(sys.argv) != 4:

print 'Usage: wlst.sh printMemoryInformation.py <wls\_url> <wls\_username> <wls\_password>';

exit();

adminURL = sys.argv[1];

adminUserName = sys.argv[2];

adminPassword = sys.argv[3];

# note: alternatively you could have used security files created by storeconfig

# connect to the admin server

connect(adminUserName, adminPassword, adminURL),

# get the server namees

serverNames = cmo.getServers()

domainRuntime()

# loop over all servers in the domain

for nextServerName in serverNames:

try:

# change to the next server JVM runtime information mbean

cd("/ServerRuntimes/"+nextServerName.getName()+"/JVMRuntime/"+nextServerName.getName())

totalMemory = get('HeapSizeCurrent')

freeMemory = get('HeapFreeCurrent')

usedMemory = (totalMemory - freeMemory)

print "Memory information for server: "+ nextServerName.getName();

print "----------------------------------------------------------------"

print 'Total memory allocated : ' + str(totalMemory)

print 'Free memory : ' + str(freeMemory)

print 'Memory in use : ' + str(usedMemory)

print '\n';

except:

dumpStack()

print 'Problem getting memory information for server '+ nextServerName.getName();

disconnect();

* monitor\_all\_destinations

# ( Script provided by Andrew Pioro )

import thread

import time

from time import gmtime, strftime

username='weblogic'

password='weblogic123'

wlsUrl='t3://127.0.0.1:7001'

connect(username,password, wlsUrl)

def getTime():

return strftime("%Y-%m-%d %H:%M:%S", gmtime())

def monitorJms():

servers = domainRuntimeService.getServerRuntimes();

if (len(servers) > 0):

for server in servers:

jmsRuntime = server.getJMSRuntime();

jmsServers = jmsRuntime.getJMSServers();

for jmsServer in jmsServers:

destinations = jmsServer.getDestinations();

for destination in destinations:

try:

print getTime() , '|' , server.getName() , '|' , jmsServer.getName() , '|' ,

destination.getName() , '|' ,destination.getMessagesCurrentCount(), '|' ,

destination.getMessagesPendingCount() , '|',

destination.getMessagesHighCount() , '|' ,

destination.getMessagesReceivedCount() , '|' ,

destination.getMessagesMovedCurrentCount() , '|' ,

destination.getConsumersCurrentCount() , '|' ,

destination.getConsumersHighCount() , '|' ,

destination.getConsumersTotalCount()

except:

print 'ERROR\_DATA';

print 'Time | ServerName | JMSServerName | DestName | MessagesCurrentCount | MessagesPendingCount | MessagesHighCount | MessagesReceivedCount | MessagesMovedCurrentCount | ConsumersCurrentCount | ConsumersHighCount | ConsumersTotalCount';

while 1:

monitorJms();

print '';

java.lang.Thread.sleep(15000);

(Script provided by Andrew Pioro)

* check\_states

import os

def sendMail(domain,count,dest,sname):

str3 = 'Domain:#' + domain +'# Server Name: '+ sname +', '+ dest +' @' + str(count)

cmd = "echo " + str3 + " > /tmp/rw\_file"

os.system(cmd)

os.system('/bin/mailx -s "Attention, The application state is abnormal !!! " operators@test.com < /tmp/rw\_file')

print '\*\*\*\*\*\*\*\*\* ALTERT MAIL HAS BEEN SENT \*\*\*\*\*\*\*\*\*\*\*'

cmd = "rm /tmp/rw\_file"

os.system(cmd)

def ServerState():

servers = getRunningServerNames()

sign="###"

activecount=0

problemcount=0

cd('domainRuntime:/AppRuntimeStateRuntime/AppRuntimeStateRuntime')

apps = getAppStatus()

for app in apps:

for server in servers:

cd("/ServerLifeCycleRuntimes/" + server.getName())

serverState = cmo.getState()

if serverState == "RUNNING" and server.getName()!="ADMINHOST" :

cd('domainRuntime:/AppRuntimeStateRuntime/AppRuntimeStateRuntime')

currentState = str(cmo.getCurrentState(app, server.getName()))

if currentState == "STATE\_ACTIVE":

print '%25s, %25s :\033[1;32m %1s %15s \033[0m' % (app,server.getName(),sign,currentState)

activecount=activecount + 1

elif currentState == "STATE\_NEW" or currentState =="None":

print '%25s, %25s :\033[1;30m %1s %15s \033[0m' % (app,server.getName(),sign,currentState)

elif currentState == "STATE\_PREPARED" or currentState == "STATE\_ADMIN" :

print '%25s, %25s :\033[1;31m %1s %15s \033[0m' % (app,server.getName(),sign,currentState)

try:

progress=startApplication(app)

except:

print 'Skipping ..'

sendMail(aserver,app,currentState,server.getName())

problemcount=problemcount +1

currentState = str(cmo.getCurrentState(app, server.getName()))

print '%25s, %25s :\033[1;30m %1s %15s \033[0m' % (app,server.getName(),sign,currentState)

else:

sendMail(aserver,app,currentState,server.getName())

problemcount=problemcount +1

print '%25s, %25s :\033[1;31m %1s %15s \033[0m' % (app,server.getName(),sign,currentState)

print 'Total %4s STATE\_ACTIVE :\033[1;31m %4s Problem \033[0m' %(repr(activecount),repr(problemcount))

exit ()

def getAppStatus():

cd('domainRuntime:/AppRuntimeStateRuntime/AppRuntimeStateRuntime')

return cmo.getApplicationIds()

def getRunningServerNames():

domainConfig()

return cmo.getServers()

if \_\_name\_\_== "main":

redirect('Server.log', 'false')

connect (str(user),str(password),str(adminserverURL))

ServerState()

(Source: Script written by Atilla Demirel)

* EJB\_print\_information

# print EJB information

def printEJBRuntimeInformation(applicationName,ejbComponentRuntime):

myName = ejbComponentRuntime.getName()

print '\nFound EJB module '+myName+' with current deployment state = ' + getApplicationComponentState(ejbComponentRuntime)

# now look at the submodules

myEJBs = ejbComponentRuntime.getEJBRuntimes()

for nextEJB in myEJBs:

nextEjbType = nextEJB.getType()

print ' Found EJB module: ' + nextEJB.getName() + ' of type ' + nextEjbType

if nextEjbType == 'EntityEJBRuntime':

# transaction information from TransactionRuntime

myTxRuntime = nextEJB.getTransactionRuntime()

print ' TransactionInfo: total:'

+ str(myTxRuntime.getTransactionsCommittedTotalCount())

+' rolledback:'+str(myTxRuntime.getTransactionsRolledBackTotalCount())

+' timedout:' + str(myTxRuntime.getTransactionsTimedOutTotalCount())

elif nextEjbType == 'StatelessEJBRuntime':

# transaction information from TransactionRuntime

myTxRuntime = nextEJB.getTransactionRuntime()

print ' TransactionInfo: total:'

+ str(myTxRuntime.getTransactionsCommittedTotalCount())

+' rolledback:'+str(myTxRuntime.getTransactionsRolledBackTotalCount())

+' timedout:'

+ str(myTxRuntime.getTransactionsTimedOutTotalCount())

# pool information from PoolRuntime

myPoolRuntime = nextEJB.getPoolRuntime()

print ' PoolInfo: totalAccess:'+str(myPoolRuntime.getAccessTotalCount())

+' beansInUse:'+str(myPoolRuntime.getBeansInUseCount())

+' beansInUseCurrent:'+str(myPoolRuntime.getBeansInUseCurrentCount())

+' destroyed:'+str(myPoolRuntime.getDestroyedTotalCount())

+' idle:'+str(myPoolRuntime.getIdleBeansCount())

+' pooledCurrent:'+str(myPoolRuntime.getPooledBeansCurrentCount())

+' timedout:'+str(myPoolRuntime.getTimeoutTotalCount())

# timer consists of a list of timers

myTimerRuntime = nextEJB.getTimerRuntime()

if myTimerRuntime != None:

print ' Timer information: Name:'+myTimerRuntime.getName()

+' activeTimers:'+str(myTimerRuntime.getActiveTimerCount())

+' timeout:'+str(myTimerRuntime.getTimeoutCount())

+' cancelled:'+str(myTimerRuntime.getCancelledTimerCount())

+' disabled:'+str(myTimerRuntime.getDisabledTimerCount())

elif nextEjbType == 'StatefulEJBRuntime':

# transaction information from TransactionRuntime

myTxRuntime = nextEJB.getTransactionRuntime()

print ' TransactionInfo: total:'

+str(myTxRuntime.getTransactionsCommittedTotalCount())

+' rolledback:'+str(myTxRuntime.getTransactionsRolledBackTotalCount())

+' timedout:' + str(myTxRuntime.getTransactionsTimedOutTotalCount())

# cache information from CacheRuntime

myCacheRuntime = nextEJB.getCacheRuntime()

print ' CacheInfo: hits:'+str(myCacheRuntime.getCacheHitCount())

+' currentBeans:'+str(myCacheRuntime.getCachedBeansCurrentCount())

+' access:'+str(myCacheRuntime.getCacheAccessCount())

# locking information from LockingRuntime

myLockingRuntime = nextEJB.getLockingRuntime()

print ' LockingInfo: currentCount:'

+ str(myLockingRuntime.getLockEntriesCurrentCount())

+' accessCount:'+str(myLockingRuntime.getLockManagerAccessCount())

+' timeoutTotalCount:'+str(myLockingRuntime.getTimeoutTotalCount())

elif nextEjbType == 'MessageDrivenEJBRuntime':

# transaction information from TransactionRuntime

myTxRuntime = nextEJB.getTransactionRuntime()

print ' TransactionInfo: total:'

+str(myTxRuntime.getTransactionsCommittedTotalCount())

+' rolledback:'+str(myTxRuntime.getTransactionsRolledBackTotalCount())

+' timedout:' + str(myTxRuntime.getTransactionsTimedOutTotalCount())

* define\_debug\_area\_functions

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

domainProps = Properties();

userConfigFile = '';

userKeyFile = '';

groupname = '';

debugvalue = '';

###################################################

# Load properties

###################################################

def intialize():

global domainProps;

global userConfigFile;

global userKeyFile;

global groupname;

global debugvalue;

# test arguments

if len(sys.argv) != 6:

print 'Usage: setdebuggroup.py <property\_file> <groupname> <debugvalue>'

exit();

try:

domainProps = Properties()

# load properties and overwrite defaults

input = FileInputStream(sys.argv[1])

domainProps.load(input)

input.close()

groupname = sys.argv[2]

debugvalue = sys.argv[3]

userConfigFile = sys.argv[4]

userKeyFile = sys.argv[5]

except:

print 'Cannot load properties !'

exit();

################################

# Connect to adminserver

################################

def connnectToAdminServer():

try:

connUri = domainProps.getProperty('adminURL')

print 'Connecting to the Admin Server ('+connUri+')'

connect(userConfigFile=userConfigFile,userKeyFile=userKeyFile,url=connUri)

print 'Connected'

except:

dumpStack();

print 'Could not connect to admin server - script will exit !'

exit();

def set\_Cluster\_DebugFlags():

set('DebugCluster, debugvalue)

set('DebugClusterAnnouncements, debugvalue)

set('DebugClusterFragments, debugvalue)

set('DebugClusterHeartbeats, debugvalue)

def set\_EJB\_DebugFlags():

set('DebugEjbCaching, debugvalue)

set('DebugEjbCmpDeployment, debugvalue)

set('DebugEjbCmpRuntime, debugvalue)

set('DebugEjbCompilation, debugvalue)

set('DebugEjbDeployment, debugvalue)

set('DebugEjbInvoke, debugvalue)

set('DebugEjbLocking, debugvalue)

set('DebugEjbMdbConnection, debugvalue)

set('DebugEjbPooling, debugvalue)

set('DebugEjbSecurity, debugvalue)

set('DebugEjbSwapping, debugvalue)

set('DebugEjbTimers, debugvalue)

def set\_JDBC\_DebugFlags():

set('DebugJDBCConn, debugvalue)

set('DebugJDBCDriverLogging, debugvalue)

set('DebugJDBCInternal, debugvalue)

set('DebugJDBCONS, debugvalue)

set('DebugJDBCRAC, debugvalue)

set('DebugJDBCREPLAY, debugvalue)

set('DebugJDBCRMI, debugvalue)

set('DebugJDBCSQL, debugvalue)

set('DebugJDBCUCP, debugvalue)

def set\_JMS\_DebugFlags():

set('DebugJMSBackEnd, debugvalue)

set('DebugJMSBoot, debugvalue)

set('DebugJMSCommon, debugvalue)

set('DebugJMSConfig, debugvalue)

set('DebugJMSDispatcher, debugvalue)

set('DebugJMSDistTopic, debugvalue)

set('DebugJMSLocking, debugvalue)

set('DebugJMSMessagePath, debugvalue)

set('DebugJMSModule, debugvalue)

set('DebugJMSPauseResume, debugvalue)

set('DebugJMSStore, debugvalue)

set('DebugJMST3Server, debugvalue)

set('DebugJMSWrappers, debugvalue)

set('DebugJMSXA, debugvalue)

def set\_JTA\_DebugFlags():

set('DebugJTA2PC, debugvalue)

set('DebugJTA2PCStackTrace, debugvalue)

set('DebugJTAAPI, debugvalue)

set('DebugJTAGateway, debugvalue)

set('DebugJTAGatewayStackTrace, debugvalue)

set('DebugJTAHealth, debugvalue)

set('DebugJTAJDBC, debugvalue)

set('DebugJTALifecycle, debugvalue)

set('DebugJTAMigration, debugvalue)

set('DebugJTANaming, debugvalue)

set('DebugJTANamingStackTrace, debugvalue)

set('DebugJTANonXA, debugvalue)

set('DebugJTAPropagate, debugvalue)

set('DebugJTARMI, debugvalue)

set('DebugJTARecovery, debugvalue)

set('DebugJTARecoveryStackTrace, debugvalue)

set('DebugJTAResourceHealth, debugvalue)

set('DebugJTAResourceName, debugvalue)

set('DebugJTATLOG, debugvalue)

set('DebugJTATransactionName, debugvalue)

set('DebugJTAXA, debugvalue)

set('DebugJTAXAStackTrace, debugvalue)

def set\_Security\_DebugFlags():

set('DebugSecurityAdjudicator, debugvalue)

set('DebugSecurityAtn, debugvalue)

set('DebugSecurityAtz, debugvalue)

set('DebugSecurityAuditor, debugvalue)

set('DebugSecurityCertPath, debugvalue)

set('DebugSecurityCredMap, debugvalue)

set('DebugSecurityEEngine, debugvalue)

set('DebugSecurityEncryptionService, debugvalue)

set('DebugSecurityJACC, debugvalue)

set('DebugSecurityJACCNonPolicy, debugvalue)

set('DebugSecurityJACCPolicy, debugvalue)

set('DebugSecurityKeyStore, debugvalue)

set('DebugSecurityPasswordPolicy, debugvalue)

set('DebugSecurityPredicate, debugvalue)

set('DebugSecurityRealm, debugvalue)

set('DebugSecurityRoleMap, debugvalue)

set('DebugSecuritySAML2Atn, debugvalue)

set('DebugSecuritySAML2CredMap, debugvalue)

set('DebugSecuritySAML2Lib, debugvalue)

set('DebugSecuritySAML2Service, debugvalue)

set('DebugSecuritySAMLAtn, debugvalue)

set('DebugSecuritySAMLCredMap, debugvalue)

set('DebugSecuritySAMLLib, debugvalue)

set('DebugSecuritySAMLService, debugvalue)

set('DebugSecuritySSL, debugvalue)

set('DebugSecuritySSLEaten, debugvalue)

set('DebugSecurityService, debugvalue)

set('DebugSecurityUserLockout, debugvalue)

if \_\_name\_\_== "main":

# connect to the server

connnectToAdminServer():

# switch to edit mode

edit()

# start editing

startEdit()

# change to the debug mbean

cd('Servers/'+serverName'/ServerDebug/'+serverName)

# change the debug value(s) you need to

if ('CLUSTER' == groupname):

set\_Cluster\_DebugFlags()

elif ('EJB' == groupname):

set\_EJB\_DebugFlags()

elif ('JDBC' == groupname):

set\_JDBC\_DebugFlags()

elif ('JMS' == groupname):

set\_JMS\_DebugFlags()

elif ('JTA' == groupname):

set\_JTA\_DebugFlags()

elif ('SECURITY' == groupname):

set\_Security\_DebugFlags()

# save and activate your changes

save()

activate()

# Good bye

exit()

* set\_desired\_values

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

if \_\_name\_\_== "main":

myUserConfigFile = sys.argv[1]

myUserKeyFile = sys.argv[2]

connUri = sys.argv[3]

debug\_props\_file = sys.argv[4]

debugProps = Properties();

print 'Reading debugflags from '+debug\_props\_file;

debugFile = FileInputStream(debug\_props\_file)

debugProps.load(debugFile)

print 'Connecting to the Admin Server ('+connUri+')';

connect(userConfigFile=myUserConfigFile,userKeyFile=myUserKeyFile,url=connUri);

edit()

startEdit()

if len(sys.argv) == 5:

# change to the debug mbean

cd('Servers/'+serverName+'/ServerDebug/'+serverName)

elif len(sys.argv) == 6:

cd('Servers/'+sys.argv[5]+'/ServerDebug/'+sys.argv[5])

else:

print 'Wrong number of properties'

exit()

# so if all ok, then set all properties

ls()

# iterator over properies

for prop in debugProps:

print 'Set '+prop+' to '+debugProps.getProperty(prop)

set(prop, debugProps.getProperty(prop))

ls()

# save and activate your changes

save()

activate()

exit()

* create\_threadDump

import sys

from java.util import Properties

from java.io import FileInputStream

from java.io import File

# ... global variables

domainProps = Properties();

userConfigFile = '';

userKeyFile = '';

overwrittenServerURL =''

###################################################################

# Load properties

###################################################################

def intialize():

global domainProps;

global userConfigFile;

global userKeyFile;

global overwrittenServerURL;

global levelToPrint;

# test arguments

if ((len(sys.argv) != 2) and (len(sys.argv) != 3)):

print 'Usage: threaddump.py <property\_file>';

print 'OR'

print 'Usage: threaddump.py <property\_file> <ADMIN-URL overwrite>';

exit();

print 'Starting the initialization process';

try:

domainProps = Properties()

# load properties and overwrite defaults

input = FileInputStream(sys.argv[1])

domainProps.load(input)

input.close()

userConfigFile = domainProps.getProperty('userConfigFile')

userKeyFile = domainProps.getProperty('userKeyFile ')

if (len(sys.argv) == 3):

overwrittenServerURL = sys.argv[2];

except:

dumpStack()

print 'Cannot load properties !';

exit();

print 'Initialization completed';

###################################################################

# Connect to server

###################################################################

def connnectToAdminServer():

# if NO other URL was provided, the default admin URL from the property file will be used

if (overwrittenServerURL==''):

connUri = domainProps.getProperty('adminURL')

else:

# Ok, overwrite, this means the URL provided as last argument will be used

connURI=overwrittenServerURL

print 'Connecting to the server ('+connUri+')';

connect(userConfigFile=userConfigFile,userKeyFile=userKeyFile,url=connUri);

print 'Connected';

if \_\_name\_\_== "main":

intialize()

connnectToAdminServer()

threadDump()

disconnect()

* threadDump\_email\_notification

import os

def atuning(sname):

t\_limit=15

cd("/ServerRuntimes/" + sname)

cd("ThreadPoolRuntime/ThreadPoolRuntime/")

pr=cmo.getPendingUserRequestCount()

ql=cmo.getQueueLength()

queuename=cmo.getName()

if pr > t\_limit :

tdump(sname,queuename,pr)

print 'ServerName %-20s# Queuename %-45s Pending Request %-4d Queue %-3d'

%(sname,queuename,repr(pr),repr(ql))

def tdump(sname,qname,qvalue):

dump\_file= '/tmp/'+sname

threadDump(writeToFile='true',fileName=dump\_file, serverName=sname)

sendMail("SHADOW",sname,qname,dump\_file,qvalue)

def sendMail(count,dest,qname,dumpfile\_tmp,qvalue):

cmd1 = "echo Queue Name:" + str(qname) + " Queue Value:" + str(qvalue) + " > rw\_tdumpfile"

cmd2 = "cat " + dumpfile\_tmp + ">> rw\_tdumpfile"

os.system(cmd1)

os.system(cmd2)

os.system('/bin/mailx -s "Achtung : You have pending requests " test@gmail.com < rw\_tdumpfile')

def monitorServerState():

connect(uname, pwd, url)

t\_limit=15

DSlist=["null"]

serverNames = getRunningServerNames()

domainRuntime()

for sname in serverNames:

cd("/ServerLifeCycleRuntimes/" + sname.getName())

serverState = cmo.getState()

if serverState == "RUNNING" :

print 'Now checking '+sname.getName()

cd("/ServerRuntimes/" + sname.getName())

if cmo.getThreadPoolRuntime() is not None:

atuning(sname.getName())

else:

equeue=cmo.getExecuteQueueRuntimes()

for names in equeue:

cd("ExecuteQueueRuntimes/" + names.getName())

executeTTC=cmo.getExecuteThreadTotalCount()

ETCIC=cmo.getExecuteThreadCurrentIdleCount()

PRCC=cmo.getPendingRequestCurrentCount()

queuename=cmo.getName()

if PRCC > t\_limit :

tdump(sname.getName(),queuename,PRCC)

print 'ServerName %-20s# Queuename %-45s \033[1;31m Pending Request %-4d \033[0m Idle Thread %3d/%-3d ' %(sname.getName(),queuename,repr(PRCC),repr(ETCIC),repr(executeTTC))

else:

print 'ServerName %-20s# Queuename %-45s Pending Request %-4d Idle Thread %3d/%-3d' %(sname.getName(),queuename,repr(PRCC),repr(ETCIC),repr(executeTTC))

cd ('../../')

cd ('../../../../')

exit()

def getRunningServerNames():

domainConfig()

return cmo.getServers()

if \_\_name\_\_== "main":

redirect('Server.log', 'false')

monitorServerState()

(Written and provided by Atilla Demirel)

* heap\_dump

connect ('weblogic','password','t3://adminhost:7700')

heap\_limit=5

import os

def sendMail(dumpfile\_tmp):

cmd = "cat " + dumpfile\_tmp + ">> rw\_file "

os.system(cmd)

os.system('/bin/mailx -s "Achtung: There was a heap problem !!! " atillademirel@gmail.com < rw\_file')

def heapdump(sname,sip):

from java.io import FileInputStream

import java.lang

import os

import string

cmd = "ssh -qn " + sip[1] +

"`` ps -ef | grep java |grep " +str(sname) +" |awk '{print pid= $2}' > /tmp/tmp1.txt``` "

os.system(cmd)

propInputStream = FileInputStream("/tmp/tmp1.txt")

configProps = Properties()

configProps.load(propInputStream)

spid = configProps.get("pid")

cmd1 = "ssh -qn " + sip[1] + "`` /data/hdump.sh "+spid+" `` "

os.system(cmd1)

def ServerState():

domainRuntime()

servers = domainRuntimeService.getServerRuntimes();

sign="%"

for server in servers:

cd("/ServerLifeCycleRuntimes/" + server.getName())

serverState = cmo.getState()

if serverState == "RUNNING" :

cd("/ServerRuntimes/" + server.getName())

sip=str(cmo.getListenAddress()).split('/')

cd("JVMRuntime/" + server.getName())

heapfree=cmo.getHeapFreePercent()

if heapfree < heap\_limit :

print '%15s :\033[1;31m %15s %1s%2d \033[0m' % (server.getName(),sip[1],sign,heapfree)

cmo.runGC()

heapfree=cmo.getHeapFreePercent()

print '%15s :\033[1;32m %1s%2d \033[0m' % (server.getName(),sign,heapfree)

if heapfree < heap\_limit :

dump\_file= '/tmp/'+server.getName()

threadDump(writeToFile='true',fileName=dump\_file, serverName=server.getName())

sendMail(dump\_file)

heapdump(server.getName(),sip)

shutdown(server.getName(),timeOut=30,force="true",block="true")

else:

print '%15s :\033[1;32m %15s %1s%2d \033[0m' % (server.getName(),sip[1],sign,heapfree)

exit ()

def getRunningServerNames():

domainConfig()

return cmo.getServers()

if \_\_name\_\_== "main":

redirect('Server.log', 'false')

ServerState()

* heap\_dump\_shell\_script

#!/bin/bash

#constants

white='\E[37;40m'

dump\_file\_dir=/data/dumps

#defult home

JAVA\_HOME=/data/jdk617/bin

JROC\_HOME=/data/jdk16/bin

TIMESTAMP=`date '+%Y\_%m\_%d\_%H.%M.%S' `

EXPECTED\_ARGS=2

E\_BADARGS=65

if [ $# -gt 2 -o $# -lt 1 ];then

ps -ef |grep weblogic.Name|grep -v grep |sort

echo -e " $white\033[41m Usage : `basename $0` pid [optional] -F $white "

echo -e " $white\033[41m Sample 1: `basename $0` 1234 $white "

echo -e " $white\033[41m Sample 2: `basename $0` 1234 -F $white "

exit $E\_BADARGS

fi

hostname=`hostname`

os=`uname`

printf "$s %s #" $TIMESTAMP $1

if [ $os == "SunOS" ];then

servername=` pargs $1 | grep "Dweblogic.Name" |grep argv | awk '{print $2}' |cut -d= -f2 `

JAVA\_HOME=`pargs $1 |grep bin |grep -v arg | awk '{print $2}' |cut -d/ -f1-5`

fi

if [ $os == "Linux" ];then

servername=` tr '\0' '\n' < /proc/$1/environ |grep SERVER\_NAME|grep -v AD |cut -d= -f2 `

JAVA\_HOME=` tr '\0' '\n' < /proc/$1/environ |grep JAVA\_HOME |cut -d= -f2 `

JAVA\_HOME=$JAVA\_HOME"/bin"

fi

temp\_dir="${dump\_file\_dir}/${servername}/${TIMESTAMP}"

echo

echo -e " $white\033[41m $servername $white "

mkdir -p ${temp\_dir}

file\_name=$temp\_dir/\_$servername.$TIMESTAMP.bin

if [ -f $JAVA\_HOME/jmap ]

then

$JAVA\_HOME/jmap $2 -dump:live,format=b,file=$file\_name $1

else

$JAVA\_HOME/jrcmd $1 hprofdump filename=$file\_name

fi

if [ $? = 0 ] ; then

echo; echo; echo "Sending mail..."

{ echo "dear friends: i got a heapdump file $servername@$hostname $file\_name "

} | mailx -s Caution: i got a heapdump file ' atillademirel@gmail.com

echo "Sending mail...: DONE!"

fi

exit 0

(Written and provided by Atilla Demirel)

* browse\_JDNI\_tree

wls:/medrec/serverConfig>

wls:/medrec/serverConfig> jndi()

wls:/medrec/jndi> ls()

dr-- MedRecServer

wls:/medrec/jndi> cd ('MedRecServer')

wls:/medrec/jndi/MedRecServer> ls()

dr-- com

dr-- ejb

dr-- javax

dr-- jdbc

dr-- mail

dr-- weblogic

-r-- medrecmedrec-facade\_jarRmiPatientFacadeBroker\_Homeweblogic.rmi.cluster.ClusterableRemoteObject

-r-- mejbmejb\_jarMejb\_EO weblogic.rmi.cluster.ClusterableRemoteObject

-r-- physicianmedrec-facade\_jarRmiPatientFacadeBroker\_Homeweblogic.rmi.cluster.ClusterableRemoteObject

wls:/medrec/jndi/MedRecServer> cd ('jdbc')

wls:/medrec/jndi/MedRecServer/jdbc> ls()

-r-- MedRecGlobalDataSourceXA weblogic.rmi.cluster.ClusterableRemoteObject

wls:/medrec/jndi/MedRecServer> cd ('javax')

wls:/medrec/jndi/MedRecServer/javax> ls()

dr-- jms

dr-- transaction

wls:/medrec/jndi/MedRecServer/javax> cd ('jms')

wls:/medrec/jndi/MedRecServer/javax/jms> ls()

-r-- QueueConnectionFactory weblogic.rmi.cluster.ClusterableRemoteObject

-r-- TopicConnectionFactory weblogic.rmi.cluster.ClusterableRemoteObject

wls:/medrec/jndi/MedRecServer/javax> cd ('transaction')

wls:/medrec/jndi/MedRecServer/javax/transaction> ls()

-r-- TransactionManager weblogic.transaction.internal.ClientTransactionManagerImpl

-r-- TransactionSynchronizationRegistry weblogic.transaction.internal.ClientTransactionManagerImpl

-r-- UserTransaction weblogic.transaction.internal.ClientTransactionManagerImpl

* initialize\_connection

/\*\*

\* Init and open connection to an MBean server

\*

\*/

private void initConnection(boolean editmode,

String url,

String username,

String password,

boolean logErrors) throws WLSAutomationException

{

try {

isEdit = editmode;

connectionuser = username;

connectionpassword = password;

System.out.println("JMXWrapper:initConnection called !");

// get protocol from URL string

protocol = url.substring(0,url.indexOf("://"));

// get hostname from URL string

hostname = url.substring(url.indexOf("//")+2, url.indexOf(":",protocol.length()+3));

// get port from URL string

portString = url.substring(url.indexOf(":",protocol.length()+3)+1,url.length());

System.out.println("JMXWrapper:initConnection: Protocol = "+protocol);

System.out.println("JMXWrapper:initConnection: Hostname = "+hostname);

System.out.println("JMXWrapper:initConnection: Port = "+portString);

Integer portInteger = Integer.valueOf(portString);

int port = portInteger.intValue();

String jndiroot = "/jndi/";

String mserver = (isEdit) ? "weblogic.management.mbeanservers.edit" :

"weblogic.management.mbeanservers.domainruntime";

JMXServiceURL serviceURL = new JMXServiceURL(protocol, hostname,

port, jndiroot + mserver);

System.out.println("URL: "+serviceURL.toString());

Hashtable h = new Hashtable();

h.put(Context.SECURITY\_PRINCIPAL, connectionuser);

h.put(Context.SECURITY\_CREDENTIALS, connectionpassword);

h.put(JMXConnectorFactory.PROTOCOL\_PROVIDER\_PACKAGES,"weblogic.management.remote");

connector = JMXConnectorFactory.connect(serviceURL, h);

connection = connector.getMBeanServerConnection();

// success

initialized = true;

}

catch (Exception ex)

{

initialized = false;

if (logErrors)

System.out.println("PROBLEM with JMXWrapper:initConnection: " +

ex.getMessage());

throw new WLSAutomationException("PROBLEM with JMXWrapper:initConnection: " +

ex.getMessage());

}

}

* set\_attribute

public void setAttribute(ObjectName name, Attribute attribute) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:setAttribute called !"); // Logfile JMX\_LAYER !!!

// Log complete JMX call

// BE CAREFUL !! as passwords will be visible !!!

StringBuffer buf = new StringBuffer();

if (isEdit)

buf.append("[EDIT-MbeanTree ] ");

else // Runtime

buf.append("[RUNTIME-MBeanTree] ");

buf.append(" : MBean="+name);

System.out.println(buf.toString());

System.out.println("SET "+attribute.getName()+" to value="+attribute.getValue());

// do INVOKE

getConnection().setAttribute(name,attribute);

}

catch (Exception ex) {

throw new WLSAutomationException("PROBLEM with JMXWrapper:setAttribute: " +

ex.getMessage());

}

}

* get\_attribute

public Object getAttribute(ObjectName name, String attributeName) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:getAttribute called !"); // Logfile JMX\_LAYER !!!

// Log complete JMX call

// BE CAREFUL !! as passwords will be visible !!!

StringBuffer buf = new StringBuffer();

if (isEdit)

buf.append("[EDIT-MBeanTree ] ");

else // Runtime

buf.append("[RUNTIME-MBeanTree] ");

buf.append(" : MBean="+name);

System.out.println(buf.toString());

System.out.println("GET Attribute="+attributeName);

// do INVOKE

return getConnection().getAttribute(name,attributeName);

}

catch (Exception ex) {

throw new WLSAutomationException("PROBLEM with JMXWrapper:getAttribute: " +

ex.getMessage());

}

}

* invoke\_operation

/\*\*

\* Invoke an MBean operation

\*/

public Object invoke(ObjectName name, String operationName, Object[] params,

String[] signature) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:invoke called !"); // Logfile JMX\_LAYER !!!

if (params==null)

params = new Object[0];

if (signature == null)

signature = new String[0];

// Log complete JMX call

// BE CAREFUL !! as passwords will be visible !!!

StringBuffer buf = new StringBuffer();

if (isEdit)

buf.append("[EDIT-MbeanTree ] ");

else // Runtime

buf.append("[RUNTIME-MbeanTree] ");

buf.append(" : MBean="+name);

System.out.println(buf.toString());

System.out.println("Operation="+operationName);

// params

buf = new StringBuffer(" Parameter: ");

for (int i=0;i<params.length;i++){

buf.append(((params[i] == null) ? "NULL" : params[i].toString()) +

"[" + signature[i] + "] ");

}

System.out.println(buf.toString());

// do INVOKE

return getConnection().invoke(name, operationName, params, signature);

}

catch (Exception ex) {

throw new WLSAutomationException("PROBLEM with JMXWrapper:invoke: " + ex.getMessage());

}

}

* start\_edit\_session

/\*\*

\* Start an edit session.

\* @return ObjectName

\* @throws WLSAutomationException

\*/

private ObjectName startEditSession() throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:startEditSession called !");

// Get the object name for ConfigurationManagerMBean.

ObjectName cfgMgr = (ObjectName) getAttribute(service,"ConfigurationManager");

// use the configuration manager in order to start an edit session

ObjectName domainConfigRoot = (ObjectName)invoke(cfgMgr,

"startEdit", new Object[] {new Integer(60000),

new Integer(120000)}, new String[] {"java.lang.Integer",

"java.lang.Integer"});

if (domainConfigRoot == null)

{

// Couldn't get the lock

throw new Exception("Somebody else is editing already");

}

System.out.println("JMXWrapper:startEditSession finished !");

return domainConfigRoot;

}

catch (Exception ex) {

System.out.println("PROBLEM with JMXWrapper:startEditSession: " + ex.getMessage(), ex);

throw new WLSAutomationException("PROBLEM with startEditSession: " + ex.getMessage());

}

}

* start\_nonedit\_session

private ObjectName startNONEditSession() throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:startNONEditSession called !");

ObjectName domainMBean =(ObjectName) getAttribute(service,"DomainConfiguration");

return domainMBean;

}

catch (Exception ex) {

System.out.println("PROBLEM with JMXWrapper:startNONEditSession: " + ex.getMessage);

throw new WLSAutomationException("PROBLEM with startNONEditSession: " + ex.getMessage());

}

}

* connect

public void connectToAdminServer(boolean edit,

String username,

String password,

String url,

boolean logConnectionProblems

) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:connectToAdminServer called !");

initConnection(edit, url, username,password,logConnectionProblems);

// service depends on edit or NONedit mode

service = new ObjectName((edit ? "com.bea:Name=EditService,Type=weblogic.management.mbeanservers.edit.EditServiceMBean"

: "com.bea:Name=DomainRuntimeService,Type=weblogic.management.mbeanservers.domainruntime.DomainRuntimeServiceMBean")

);

}

catch (Exception ex) {

throw new WLSAutomationException("PROBLEM with connectToAdminServer: " + ex.getMessage());

}

}

* activate\_changes

private ObjectName activate() throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:activate called !");

ObjectName cfgMgr = (ObjectName) getAttribute(service,"ConfigurationManager");

Object o = invoke(cfgMgr, "save", null, null);

ObjectName result = (ObjectName) invoke(cfgMgr,"activate",

new Object[] {new Long(120000)}, new String[] {"java.lang.Long"});

return result;

}

catch (Exception ex)

{

System.out.println("PROBLEM with JMXWrapper:activate: " + ex.getMessage());

throw new WLSAutomationException("PROBLEM with JMXWrapper:activate: " + ex.getMessage());

}

}

* disconnect\_adminserver

public void disconnectFromAdminServer(boolean edit) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:disconnectFromAdminServer called !");

if (edit)

activate();

// Close the connection with the MBean server.

domainConfigRoot = null;

try {

connector.close();

}

catch (Exception ex) {

// ignore

}

connector = null;

}

catch (Exception ex) {

System.out.println("PROBLEM with disconnectFromAdminServer: " + ex.getMessage());

throw new WLSAutomationException("PROBLEM with disconnect: " + ex.getMessage());

}

}

* domain\_information

public Hashtable<String,String> getMainServerDomainValues() throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:getMainServerDomainValues called !");

Hashtable<String,String> result = new Hashtable<String,String>();

ObjectName domainMBean =(ObjectName) getAttribute(service,"DomainConfiguration");

String adminServerName = (String) getAttribute(domainMBean,"AdminServerName");

String domainName = domainMBean.getKeyProperty("Name");

String domainRoot = (String) getAttribute(domainMBean,"RootDirectory");

String configurationVersion = (String) getAttribute(domainMBean,"ConfigurationVersion");

result.put("adminServerName",adminServerName);

result.put("domainName",domainName);

result.put("domainRoot",domainRoot);

result.put("domainBase",domainRoot.substring(0,domainRoot.length()-(domainName.length()+1)));

result.put("configurationVersion",configurationVersion);

return result;

}

catch (Exception ex) {

System.out.println("PROBLEM with getMainServerDomainValues: " + ex.getMessage(), ex);

throw new WLSAutomationException("PROBLEM with getDomainValues: " + ex.getMessage());

}

}

* get\_JMX\_objectnames

public ArrayList<ObjectName> getManagedServerObjectNames() throws WLSAutomationException

{

try {

ArrayList<ObjectName> result = new ArrayList<ObjectName>();

System.out.println("JMXWrapper:getManagedServerObjectNames called !");

ObjectName domainMBean =(ObjectName) getAttribute(service,"DomainConfiguration");

String adminServerName = (String) getAttribute(domainMBean,"AdminServerName");

ObjectName[] serverRuntimes = (ObjectName[])getAttribute(domainMBean,"Servers");

for (int i = 0; i < serverRuntimes.length; i++)

{

String tmp = serverRuntimes[i].getKeyProperty("Name");

if (!tmp.equals(adminServerName))

result.add(serverRuntimes[i]);

}

return result;

}

catch (Exception ex)

{

System.out.println("PROBLEM with getManagedServerObjectNames: " + ex.getMessage(), ex);

throw new WLSAutomationException("PROBLEM with getMSObjectNames: " + ex.getMessage());

}

}

* server\_status

public String getServerState(String serverName) throws WLSAutomationException

{

try {

System.out.println("JMXWrapper:getServerState called !");

ObjectName serverRuntimeObjectName = new ObjectName("com.bea:Name=" + serverName +

",Type=ServerLifeCycleRuntime");

return (String)getAttribute(serverRuntimeObjectName,"State");

}

catch (Exception ex)

{

System.out.println("PROBLEM with JMXWrapper:getServerState: " + ex.getMessage());

return "\_UNKNOWN\_ERROR\_";

}

}

* open\_multiple\_connections

public class ParallelAccessTest {

public static void main(String[] args)

{

JMXThreadTest myJMXThreadTest;

try {

// create globalProperties from File

Properties globalProperties = new Properties();

globalProperties.load(new FileInputStream(args[0]));

int counter=1;

while (globalProperties.getProperty("domain."+counter+".name") != null)

{

try {

myJMXThreadTest =

new JMXThreadTest(counter,

globalProperties.getProperty("domain."+counter+".url"),

globalProperties.getProperty("domain."+counter+".user"),

globalProperties.getProperty("domain."+counter+".password"),

Integer.parseInt(globalProperties.getProperty(

("domain."+counter+"reloadinterval","3"))

);

myJMXThreadTest.start();

}

catch (Exception ex) {

ex.printStackTrace();

}

counter++;

}

}

catch (Exception ex) {

ex.printStackTrace();

}

}

}

* multi-threading\_example

import java.util.\*;

import javax.management.MBeanServerConnection;

import javax.management.ObjectName;

import javax.management.remote.\*;

import javax.naming.Context;

import java.io.FileOutputStream;

public class JMXThreadTest extends Thread {

// active (if any) mbean server connection

private MBeanServerConnection connection;

// connector

private JMXConnector connector;

private ObjectName domainRuntimeService = null;

private ObjectName domainMBean = null;

private int internalReloadInterval = 0;

private int id;

private boolean endOfLoop = false;

public JMXThreadTest(int \_id, String url, String username, String password, int reloadInterval)

{

try {

id = \_id;

internalReloadInterval = reloadInterval;

// get protocol from URL string

String protocol = url.substring(0,url.indexOf("://"));

// get hostname from URL string

String hostname = url.substring(url.indexOf("//")+2,

url.indexOf(":",protocol.length()+3));

// get port from URL string

String portString = url.substring(url.indexOf(":",protocol.length()+3)+1,url.length());

Integer portInteger = Integer.valueOf(portString);

int port = portInteger.intValue();

JMXServiceURL serviceURL = new JMXServiceURL(protocol, hostname, port,

"/jndi/weblogic.management.mbeanservers.domainruntime");

Hashtable h = new Hashtable();

h.put(Context.SECURITY\_PRINCIPAL, username);

h.put(Context.SECURITY\_CREDENTIALS, password);

h.put(JMXConnectorFactory.PROTOCOL\_PROVIDER\_PACKAGES,"weblogic.management.remote");

connector = JMXConnectorFactory.connect(serviceURL, h);

connection = connector.getMBeanServerConnection();

domainRuntimeService = new ObjectName("com.bea:Name=DomainRuntimeService,Type=weblogic.management.mbeanservers.domainruntime.DomainRuntimeServiceMBean");

domainMBean =(ObjectName)

connection.getAttribute(domainRuntimeService,"DomainConfiguration");

}

catch (Exception ex) {

ex.printStackTrace();

}

}

public void run() {

while(!endOfLoop)

{

getServersInformation();

try {

Thread.currentThread().sleep(internalReloadInterval\*1000);

}

catch (Exception ex) {

// ignore

}

}

}

public void disconnectFromAdminServer()

{

try {

endOfLoop = true;

connector.close();

connector = null;

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

public void getServersInformation()

{

try

{

FileOutputStream os = new FileOutputStream(domainMBean.getKeyProperty("Name")+".log",true);

try {

String domainName = (String) connection.getAttribute(domainMBean,"Name");

System.out.println("Aufruf Admin-Server Nr.: "+id+" : "+domainName);

os.write(("\n"+"Domain-Name : " + domainName).getBytes() );

os.write(("\n"+" AdminServerName : "+(String)

connection.getAttribute(domainMBean,"AdminServerName")).getBytes());

os.write(("\n"+" RootDirectory : "+(String)

connection.getAttribute(domainMBean,"RootDirectory")).getBytes() );

os.write(("\n"+" ProductionModeEnabled : "+((Boolean)

connection.getAttribute(domainMBean,"ProductionModeEnabled")).

booleanValue() ).getBytes());

// Managed Server

os.write(("\n"+" Managed-Server:").getBytes());

String adminServerName = (String)

connection.getAttribute(domainMBean,"AdminServerName");

ObjectName[] serverRuntimes = (ObjectName[])connection.

getAttribute(domainMBean,"Servers");

for (int i = 0; i < serverRuntimes.length; i++) {

String ms\_name = (String) connection.getAttribute(serverRuntimes[i],"Name");

if (!ms\_name.equals(adminServerName)) {

os.write(("\n"+" -> "+ms\_name + " : StartupMode = "+(String)

connection.getAttribute(serverRuntimes[i],"StartupMode")).getBytes());

os.write(("\n"+" "+ (String)connection.getAttribute(serverRuntimes[i],

"ListenAddress")+":"+((Integer)connection.

getAttribute(serverRuntimes[i],"ListenPort")).intValue() ).getBytes());

}

}

// Cluster

os.write(("\n"+" Cluster:").getBytes());

ObjectName[] cluster = (ObjectName[])connection.getAttribute(domainMBean,"Clusters");

for (int i = 0; i < cluster.length; i++)

os.write(("\n"+" -> "+(String)

connection.getAttribute(cluster[i],"Name")).getBytes());

// Machines

os.write(("\n"+" Machines:").getBytes());

ObjectName[] machines = (ObjectName[])connection.getAttribute(domainMBean,"Machines");

for (int i = 0; i < machines.length; i++)

os.write(("\n"+" -> "+(String)

connection.getAttribute(machines[i],"Name")).getBytes());

// AppDeployments

os.write(("\n"+" AppDeployments:").getBytes());

ObjectName[] appDeployments = (ObjectName[])

connection.getAttribute(domainMBean,"AppDeployments");

for (int i = 0; i < appDeployments.length; i++)

os.write(("\n"+" -> "+(String)

connection.getAttribute(appDeployments[i],"Name")).getBytes());

}

catch (Exception ex) {

os.write(("Probleme mit Admin-Server Nr.: "+id).getBytes());

os.write(ex.getMessage().getBytes());

disconnectFromAdminServer();

}

os.close();

}

catch (Exception ex) {

System.out.println("Probleme mit Admin-Server Nr.: "+id);

ex.printStackTrace();

disconnectFromAdminServer();

}

}

}

* list\_available\_MBeans

import java.util.\*;

import javax.management.\*;

import javax.management.ObjectName;

import de.wls\_monitoringbook.jmx\_access.JMXWrapper;

public class PrintMBeanNames {

public static void main(String[] args) {

System.out.println("RUNTIME:");

System.out.println("-------------------------------------------------");

printMBeans(false);

System.out.println("\n\n\n\nEDIT-MBEANServer:");

System.out.println("-------------------------------------------------");

printMBeans(true);

}

/\*\*

\* Print all MBean information from an MBean server

\* @param edit boolean

\*/

private static void printMBeans(boolean edit) {

try {

// adminUser / password / and URL should be provided as commandline parameters

String adminUser = "weblogic";

String adminPW = "Weblogic1";

String adminURL = "t3://localhost:7001";

// init wrapper

JMXWrapper myJMXWrapper = new JMXWrapper();

// connecto server - either runtime or edit mbean server

myJMXWrapper.connectToAdminServer(edit, adminUser, adminPW,adminURL, true);

MBeanServerConnection connection = myJMXWrapper.getConnection();

Set mySet = connection.queryNames(new ObjectName("\*:\*"), null);

Iterator it = mySet.iterator();

while (it.hasNext()) {

ObjectName myName = (ObjectName) it.next();

try {

System.out.println("--> " + myName.getCanonicalName());

// get all attributes

MBeanAttributeInfo[] atribs = connection.getMBeanInfo(myName).getAttributes();

for (int i = 0; i < atribs.length; i++) {

System.out.println(" Attribute: " + atribs[i].getName() +

" of Type : " + atribs[i].getType());

}

// get all operations

MBeanOperationInfo[] operations =

connection.getMBeanInfo(myName).getOperations();

for (int i = 0; i < operations.length; i++) {

System.out.print(" Operation: " +

operations[i].getReturnType() + " " +

operations[i].getName() + "(");

for (int j = 0; j < operations[i].getSignature().length;j++)

System.out.print(operations[i].getSignature()[j].

getName() + ":" +

operations[i].getSignature()[j].

getType() + " ");

System.out.println(")");

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

// disconnect

myJMXWrapper.disconnectFromAdminServer(edit);

}

catch (Exception ex) {

ex.printStackTrace();

}

}

}

* search\_MBeans

import java.io.IOException;

import java.net.MalformedURLException;

import java.util.\*;

import javax.management.MBeanServerConnection;

import javax.management.MalformedObjectNameException;

import javax.management.ObjectName;

import javax.management.remote.JMXConnector;

import javax.management.remote.JMXConnectorFactory;

import javax.management.remote.JMXServiceURL;

import javax.naming.Context;

import java.lang.\*;

public class WLSQueryTest

{

private static MBeanServerConnection connection = null;

private static JMXConnector connector = null;

private static ObjectName service = null;

public static void initConnection(String hostname, String portString,String username, String password) throws Exception

{

String protocol="t3";

Integer portInteger=Integer.valueOf(portString);

int port=portInteger.intValue();

String jndiroot="/jndi/";

String mserver=null;

String myService=System.getProperty("SERVICE","DOMAIN"); // possibe: DOMAIN, RUNTIME, EDIT

if ("DOMAIN".equalsIgnoreCase(myService))

{

service=new ObjectName("com.bea:Name=DomainRuntimeService,Type=weblogic.management.mbeanservers.domainruntime.DomainRuntimeServiceMBean");

mserver="weblogic.management.mbeanservers.domainruntime";

}

if ("RUNTIME".equalsIgnoreCase(myService))

{

service=new ObjectName("com.bea:Name=RuntimeService,Type=weblogic.management.mbeanservers.runtime.RuntimeServiceMBean");

mserver="weblogic.management.mbeanservers.runtime";

}

if ("EDIT".equalsIgnoreCase(myService))

{

service=new ObjectName("com.bea:Name=EditService,Type=weblogic.management.mbeanservers.edit.EditServiceMBean");

mserver="weblogic.management.mbeanservers.edit";

}

System.out.println(" service = " + service);

System.out.println(" JNDI = " + jndiroot + mserver);

System.out.println();

JMXServiceURL serviceURL=new JMXServiceURL(protocol, hostname,port, jndiroot + mserver);

Hashtable h=new Hashtable();

h.put(Context.SECURITY\_PRINCIPAL, username);

h.put(Context.SECURITY\_CREDENTIALS, password);

h.put(JMXConnectorFactory.PROTOCOL\_PROVIDER\_PACKAGES,"weblogic.management.remote");

connector=JMXConnectorFactory.connect(serviceURL, h);

connection=connector.getMBeanServerConnection();

}

public static void main(String[] args) throws Exception

{

String hostname=args[0];

String portString=args[1];

String username=args[2];

String password=args[3];

String queryArgument = args[4];

System.out.println("Hostname : " + hostname);

System.out.println("PortString : " + portString);

System.out.println("Username : " + username);

System.out.println("Password : " + password);

System.out.println("queryArgument : " + queryArgument);

initConnection(hostname, portString, username, password);

// query and print result

Set<ObjectName> beanSet = connection.queryNames(new ObjectName(queryArgument),null );

System.out.println("\n\nQuery found "+beanSet.size()+" results !");

ObjectName queryName = null;

Iterator<ObjectName> i = beanSet.iterator();

while (i.hasNext())

{

queryName = (ObjectName)i.next();

System.out.println(" --> " + queryName);

}

connector.close();

}

* domain\_log\_change\_jmx

Mbean needed for the configuration of the domain log file. The Mbean is shown here in order to give you an impression about the detailed control which is offered my the WebLogic management APIs

com.bea:Name=TestDomain,Type=Log

Attribute: RotateLogOnStartup of Type : java.lang.Boolean

Attribute: DomainLogBroadcastFilter of Type : javax.management.ObjectName

Attribute: RotationTime of Type : java.lang.String

Attribute: MemoryBufferSize of Type : java.lang.Integer

Attribute: Type of Type : java.lang.String

Attribute: LoggerSeverity of Type : java.lang.String

Attribute: StdoutFormat of Type : java.lang.String

Attribute: BufferSizeKB of Type : java.lang.Integer

Attribute: ServerLoggingBridgeUseParentLoggersEnabled of Type : java.lang.Boolean

Attribute: RedirectStderrToServerLogEnabled of Type : java.lang.Boolean

Attribute: StdoutSeverity of Type : java.lang.String

Attribute: DomainLogBroadcastSeverity of Type : java.lang.String

Attribute: MemoryBufferSeverity of Type : java.lang.String

Attribute: StdoutLogStack of Type : java.lang.Boolean

Attribute: RedirectStdoutToServerLogEnabled of Type : java.lang.Boolean

Attribute: StdoutFilter of Type : javax.management.ObjectName

Attribute: Parent of Type : javax.management.ObjectName

Attribute: FileTimeSpan of Type : java.lang.Integer

Attribute: LogFileRotationDir of Type : java.lang.String

Attribute: LoggerSeverityProperties of Type : java.util.Properties

Attribute: NumberOfFilesLimited of Type : java.lang.Boolean

Attribute: FileCount of Type : java.lang.Integer

Attribute: RotationType of Type : java.lang.String

Attribute: DomainLogBroadcasterBufferSize of Type : java.lang.Integer

Attribute: LogFileFilter of Type : javax.management.ObjectName

Attribute: Notes of Type : java.lang.String

Attribute: Name of Type : java.lang.String

Attribute: MemoryBufferFilter of Type : javax.management.ObjectName

Attribute: Log4jLoggingEnabled of Type : java.lang.Boolean

Attribute: FileMinSize of Type : java.lang.Integer

Attribute: FileName of Type : java.lang.String

Attribute: StacktraceDepth of Type : java.lang.Integer

Attribute: DateFormatPattern of Type : java.lang.String

Attribute: LogFileSeverity of Type : java.lang.String

* domain\_log\_example

import java.util.\*;

import javax.management.MBeanServerConnection;

import javax.management.ObjectName;

import javax.management.Attribute;

import javax.management.remote.\*;

import de.wls\_monitoringbook.jmx\_access.JMXWrapper;

public class DomainConfiguration

{

private JMXWrapper myJMXWrapper;

private Hashtable<String , String> domainInfo;

private String domainName = "";

public DomainConfiguration(JMXWrapper \_myJMXWrapper) throws Exception {

myJMXWrapper = \_myJMXWrapper;

// get domainInfo

domainInfo = myJMXWrapper.getMainServerDomainValues();

domainName = domainInfo.get("domainName");

}

/\*\*

\* Configure the attribute and rollover definition of the domain log file

\* @throws Exception

\*/

public void configureDomainLog() throws Exception {

// e.g.: com.bea:Name=TestDomain,Type=Log

ObjectName myLogMBean = new ObjectName("com.bea:Name="+domainName+",Type=Log");

// set attribute RotationType

myJMXWrapper.setAttribute(myLogMBean,new Attribute("RotationType",new String("byTime")));

// set attribute RotationTime

myJMXWrapper.setAttribute(myLogMBean,new Attribute("RotationTime",new String("02:00")));

// set attribute FileTimeSpan

myJMXWrapper.setAttribute(myLogMBean,new Attribute("FileTimeSpan",new Integer(24)));

// set attribute NumberOfFilesLimited

myJMXWrapper.setAttribute(myLogMBean,new Attribute("NumberOfFilesLimited",

new Boolean(true)));

// set attribute FileCount

myJMXWrapper.setAttribute(myLogMBean,new Attribute("FileCount",new Integer(14))); // 2 weeks

// set attribute FileName

myJMXWrapper.setAttribute(myLogMBean,

new Attribute("FileName","/applications/log/"+domainName+"domainLog.log"));

}

/\*\*

\* Configure various settings of the domain.

\* @throws Exception

\*/

public void configureDomain() throws Exception {

configureDomainLog();

}

public static void main(String[] args) throws Exception

{

JMXWrapper myJMXWrapper = new JMXWrapper();

// opens an connection to the "EDIT" MBean tree

myJMXWrapper.connectToAdminServer(true, // edit

"weblogic",

"test1234",

"t3://localhost:7001",

true

);

// create configuration instance

DomainConfiguration myDomainConfiguration = new DomainConfiguration(myJMXWrapper);

// configure domain

myDomainConfiguration.configureDomain();

// activate changes and close the connection

myJMXWrapper.disconnectFromAdminServer(true);

}

}

* configure\_server\_logs\_jmx

/\*\*

\* Configure the attribute and rollover definition of a server log file

\* @throws Exception

\*/

public void configureServerLog(String servername) throws Exception

{

// e.g.: com.bea:Name=AdminServer,Server=AdminServer,Type=Log

ObjectName myLogMBean = new ObjectName("com.bea:Name="+servername+

",Server="+servername+",Type=Log");

// e.g.: com.bea:Name=AdminServer,Server=AdminServer,Type=WebServerLog,WebServer=AdminServer

ObjectName myWebserverLogMBean = new ObjectName("com.bea:Name="+servername+",

Server="+servername+",Type=WebServerLog,WebServer="+servername+"");

// set attribute RotationType

myJMXWrapper.setAttribute(myLogMBean,new Attribute("RotationType",new String("byTime")));

// set attribute RotationTime

myJMXWrapper.setAttribute(myLogMBean,new Attribute("RotationTime",new String("02:00")));

// set attribute FileTimeSpan

myJMXWrapper.setAttribute(myLogMBean,new Attribute("FileTimeSpan",new Integer(24)));

// set attribute NumberOfFilesLimited

myJMXWrapper.setAttribute(myLogMBean,new Attribute("NumberOfFilesLimited",

new Boolean(true)));

// set attribute FileCount

myJMXWrapper.setAttribute(myLogMBean,new Attribute("FileCount",new Integer(14))); // 2 weeks

// set attribute FileName

myJMXWrapper.setAttribute(myLogMBean,

new Attribute("FileName","/applications/log/"+

servername+"/"+servername+".log"));

// now do the same for the access log of the webserver

// set attribute RotationType

myJMXWrapper.setAttribute(myWebserverLogMBean,new Attribute("RotationType",

new String("byTime")));

// set attribute RotationTime

myJMXWrapper.setAttribute(myWebserverLogMBean,new Attribute("RotationTime",

new String("02:00")));

// set attribute FileTimeSpan

myJMXWrapper.setAttribute(myWebserverLogMBean,new Attribute("FileTimeSpan",new Integer(24)));

// set attribute NumberOfFilesLimited

myJMXWrapper.setAttribute(myWebserverLogMBean,new Attribute("NumberOfFilesLimited",

new Boolean(true)));

// set attribute FileCount

myJMXWrapper.setAttribute(myWebserverLogMBean,new Attribute("FileCount",new Integer(14)));

// set attribute FileName

myJMXWrapper.setAttribute(myWebserverLogMBean,

new Attribute("FileName","/applications/log/"+

servername+"/access.log"));

}

* add\_managed-server\_jmx

public void createManagedServer(String servername, String clustername, String machinename)

throws Exception

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = new ObjectName("com.bea:Name=" + domainName +",Type=Domain");

// javax.management.ObjectName createCluster(name:java.lang.String )

// the new mbean will have a name similiar to com.bea:Name=TestManagdServer,Type=Server

ObjectName myManagedServerMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createServer",

new Object[]{new String(servername)},

new String[]{String.class.getName()});

// and SSL: com.bea:Name=TestManagdServer,Server=TestManagdServer,Type=SSL

ObjectName myManagedServerSSLMBean = new ObjectName("com.bea:Name="+servername+",Server="+servername+",Type=SSL");

// and ServerStart: com.bea:Name=MyTestServer,Server=MyTestServer,Type=ServerStart

ObjectName myManagedServerServerStartMBean =

new ObjectName("com.bea:Name="+servername+",Server="+servername+",Type=ServerStart");

// set attribute ListenAddress

myJMXWrapper.setAttribute(myManagedServerMBean,

new Attribute("ListenAddress",

new String("myManagedServerListenerAddress")));

// set attribute ListenPort

myJMXWrapper.setAttribute(myManagedServerMBean,new Attribute("ListenPort",new Integer(10432)));

// set attribute ListenPortEnabled

myJMXWrapper.setAttribute(myManagedServerMBean,new Attribute("ListenPortEnabled",new Boolean(true)));

// add to machine

if (machinename != null)

myJMXWrapper.setAttribute(myManagedServerMBean,

new Attribute("Machine",new ObjectName("com.bea:Name="+machinename+",Type=UnixMachine")));

// add to cluster

// set attribute Cluster

if (clustername != null)

myJMXWrapper.setAttribute(myManagedServerMBean,new Attribute("Cluster",

new ObjectName("com.bea:Name="+clustername+",Type=Cluster")));

// SSL

// set attribute ListenPort

myJMXWrapper.setAttribute(myManagedServerSSLMBean,new Attribute("ListenPort",new Integer(11111)));

// set attribute Enabled

myJMXWrapper.setAttribute(myManagedServerSSLMBean,new Attribute("Enabled",new Boolean(true)));

// SERVERSTART

// note the following paths and file names are only examples !!

String ms\_out = "/logs/domains/"+domainName+"/"+servername+"/"+servername+".out";

String ms\_err = "/logs/domains/"+domainName+"/"+servername+"/"+servername+".err";

String managedserver\_args = "-Djava.awt.headless=true -Xmx1024 -XX:MaxPermSize=512m -Dweblogic.Stdout="+ms\_out+" -Dweblogic.Stderr="+ ms\_err;

// set attribute Arguments

myJMXWrapper.setAttribute(myManagedServerServerStartMBean,new Attribute("Arguments",managedserver\_args));

// set attribute JavaHome

myJMXWrapper.setAttribute(myManagedServerServerStartMBean,new Attribute("JavaHome","/opt/jdks/jdk1.6"));

}

* renaming\_machine\_jmx

public void renameMachineName() throws WLSAutomationException

{

// change machine names

try

{

// get domain configuration mbean

ObjectName domainMBean =(ObjectName)myJMXWrapper.getAttribute(

myJMXWrapper.getService(),"DomainConfiguration");

// get all machines

ObjectName[] old\_machines = (ObjectName[])myJMXWrapper.getAttribute(domainMBean,"Machines");

// a list of all machines in a hasmap format for better mapping

HashMap<String,ObjectName> allMachines = new HashMap<String,ObjectName>();

// a list of all machine names for better mapping and increased performance

ArrayList<String> allMachineNames = new ArrayList<String>();

// iterate over all old machine and create the two lists defined

for (int om=0;om<old\_machines.length;om++)

{

String nextname = (String)myJMXWrapper.getAttribute(old\_machines[om],"Name");

System.out.println(" Found machine "+nextname+" of type "+old\_machines[om].toString());

allMachines.put(nextname,old\_machines[om]);

allMachineNames.add(nextname);

}

// a list to collect all machines which can be deleted

ArrayList<ObjectName> toBeDeletedMachinesMap = new ArrayList<ObjectName>();

// a list with all new machines

HashMap<String,ObjectName> myNewMachines = new HashMap<String,ObjectName>();

// iterate over all machines

Iterator omIt = allMachines.keySet().iterator();

while (omIt.hasNext())

{

String nextOldName = (String)omIt.next();

ObjectName nextOldMachine = (ObjectName)allMachines.get(nextOldName);

System.out.println("Testing machine: "+nextOldName+

" with objectname="+nextOldMachine.toString());

// This is only an example in order to demo this functionality.

// normally a mapping method which returns the newName if a new name is required

// String newName = getNewNameIfMachineMustBeRenamed(nextOldName);

// for demo purpose change all machines and add\_TEST to the machine name

String newName = nextOldName+"\_MyTEST";

if (newName == null)

{ // for the demo this code will never be reached !!

System.out.println("Machine "+nextOldName+" MUST NOT be converted !");

}

else // ok something to do

{

System.out.println("New Name for machine "+nextOldName+" is "+newName);

// create Machine

Object values[] = { newName};

String signatur[] = { String.class.getName()};

if (allMachines.containsKey(newName))

{

System.out.println("??? Machine "+newName+" ALREADY EXISTS - using this one !!");

toBeDeletedMachinesMap.add(nextOldMachine); // this machine will later be deleted !!

myNewMachines.put(nextOldName,(ObjectName)allMachines.get(newName));

}

else // create a new machine and then copy all values (including nodemanager !!)

{

System.out.println("Try to create machine "+newName);

ObjectName newMachine = (ObjectName)myJMXWrapper.invoke(domainMBean,

"createUnixMachine",values,signatur);

System.out.println("Created machine"+newName);

toBeDeletedMachinesMap.add(nextOldMachine); // this machine will later be deleted !!

myNewMachines.put(nextOldName,newMachine);

System.out.println("Copy nodemanager from machine "+nextOldName+" to "+newName);

// old nodemanager

ObjectName myOldNodeManager = (ObjectName)myJMXWrapper.getAttribute(nextOldMachine,

"NodeManager");

// new nodemanager

ObjectName myNewNodeManager = (ObjectName)myJMXWrapper.getAttribute(newMachine,

"NodeManager");

// copy all values from old machine to new machine

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("NMType",myJMXWrapper.getAttribute(myOldNodeManager,

"NMType")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("NodeManagerHome",myJMXWrapper.getAttribute(myOldNodeManager,

"NodeManagerHome")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("ListenPort",myJMXWrapper.getAttribute(myOldNodeManager,

"ListenPort")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("Notes",myJMXWrapper.getAttribute(myOldNodeManager,"Notes")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("Name",myJMXWrapper.getAttribute(myOldNodeManager,"Name")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("DebugEnabled",myJMXWrapper.getAttribute(myOldNodeManager,

"DebugEnabled")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("ShellCommand",myJMXWrapper.getAttribute(myOldNodeManager,

"ShellCommand")));

myJMXWrapper.setAttribute(myNewNodeManager,

new Attribute("ListenAddress",myJMXWrapper.getAttribute(myOldNodeManager,

"ListenAddress")));

}

}

}

// now as all new machines are created, reconfigure - means retarget the appropriate server

System.out.println("RECONFIGURE all servers if necessary !");

ObjectName[] serverRuntimes = (ObjectName[])myJMXWrapper.getAttribute(domainMBean,"Servers");

for (int i = 0; i < serverRuntimes.length; i++)

{

String serverName = serverRuntimes[i].getKeyProperty("Name");

System.out.println("Try to reconfigure "+serverName);

// Machine

try {

// get the machine of this managed server , if any

ObjectName server\_machine\_object = (ObjectName)myJMXWrapper.

getAttribute(serverRuntimes[i],"Machine");

// if this server is attached to a machine (note adminservers often are not)

if (server\_machine\_object!=null)

{

String server\_machine\_name = (String)myJMXWrapper.

getAttribute(server\_machine\_object,"Name");

// map name and set new machine

ObjectName myNewMachineTmpName =

(ObjectName)myNewMachines.get(server\_machine\_name);

// if no entry is found in the new machine list - no new machine has been created

if (myNewMachineTmpName == null)

System.out.println("Server" +serverName+" does NOT need any modifications !");

else

{

// set Machine - re-target the server to the new machine

Attribute newMachineAttribute = new Attribute("Machine",myNewMachineTmpName);

myJMXWrapper.setAttribute(serverRuntimes[i],newMachineAttribute);

}

}

else

System.out.println("NO machine found for Server = "+serverName);

}

catch (Exception ex) {

System.out.println("Problem in SwitchAllMachinesStep:executeStep "+ex);

throw new WLSAutomationException(ex.getMessage());

}

}

// destroy all old Machine

System.out.println("DESTROY all old machines");

for (int om=0;om<toBeDeletedMachinesMap.size();om++)

{

System.out.println("DESTROY "+((ObjectName)toBeDeletedMachinesMap.get(om)).toString());

Object values[] = { (ObjectName)toBeDeletedMachinesMap.get(om)};

String signatur[] = { javax.management.ObjectName.class.getName()};

myJMXWrapper.invoke(domainMBean,"destroyMachine",values,signatur);

}

}

catch (Exception ex) {

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_network\_channel\_jmx

public void createHTTPNetworkChannel(String serverName, String channelName, int port) throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// create new channel with the provided name

ObjectName myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"lookupNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

if (myNewChannel != null)

throw new WLSAutomationException("NetworkChannel "+channelName+" already exists !");

// now create

myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"createNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

// set protocol to http

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Protocol",new String("http")));

// set listener port

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ListenPort",new Integer(port)));

// enable

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Enabled",new Boolean(true)));

// enable HttpEnabledForThisProtocol

myJMXWrapper.setAttribute(myNewChannel,new Attribute("HttpEnabledForThisProtocol",

new Boolean(true)));

// set OutboundEnabled

myJMXWrapper.setAttribute(myNewChannel,new Attribute("OutboundEnabled",new Boolean(false)));

// disable https

myJMXWrapper.setAttribute(myNewChannel,new Attribute("TwoWaySSLEnabled",new Boolean(false)));

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ClientCertificateEnforced",

new Boolean(false)));

}

catch(Exception ex)

{

throw new WLSAutomationException("Error while createHTTPNetworkChannel ("+channelName+"): "+

ex.getMessage());

}

}

* create\_T3\_channel\_jmx

public void createT3NetworkChannel(String serverName, String channelName, int port) throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// create new channel with the provided name

ObjectName myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"lookupNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

if (myNewChannel != null)

throw new WLSAutomationException("NetworkChannel "+channelName+" already exists !");

// now create

myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"createNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

// set protocol to http

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Protocol",new String("t3")));

// set listener port

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ListenPort",new Integer(port)));

// enable

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Enabled",new Boolean(true)));

// enable HttpEnabledForThisProtocol

myJMXWrapper.setAttribute(myNewChannel,new Attribute("HttpEnabledForThisProtocol",

new Boolean(false)));

// set OutboundEnabled

myJMXWrapper.setAttribute(myNewChannel,new Attribute("OutboundEnabled",new Boolean(false)));

// disable https

myJMXWrapper.setAttribute(myNewChannel,new Attribute("TwoWaySSLEnabled",new Boolean(false)));

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ClientCertificateEnforced",

new Boolean(false)));

}

catch(Exception ex) {

throw new WLSAutomationException("Error while createT3NetworkChannel ("+channelName+"): "+ ex.getMessage());

}

}

* create\_CORBA\_channel\_jmx

public void createIIOPNetworkChannel(String serverName, String channelName, int port) throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// create new channel with the provided name

ObjectName myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"lookupNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

if (myNewChannel != null)

throw new WLSAutomationException("NetworkChannel "+channelName+" already exists !");

// now create

myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"createNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

// set protocol to http

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Protocol",new String("iiop")));

// set listener port

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ListenPort",new Integer(port)));

// enable

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Enabled",new Boolean(true)));

// enable HttpEnabledForThisProtocol

myJMXWrapper.setAttribute(myNewChannel,new Attribute("HttpEnabledForThisProtocol",

new Boolean(false)));

// set OutboundEnabled

myJMXWrapper.setAttribute(myNewChannel,new Attribute("OutboundEnabled",new Boolean(false)));

// disable https

myJMXWrapper.setAttribute(myNewChannel,new Attribute("TwoWaySSLEnabled",new Boolean(false)));

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ClientCertificateEnforced",

new Boolean(false)));

}

catch(Exception ex) {

throw new WLSAutomationException("Error while createIIOPNetworkChannel ("+channelName+"): "+

ex.getMessage());

}

}

* create\_HTTPS\_channel\_jmx

public void createHTTPSNetworkChannel(String serverName, String channelName, int port, boolean enforceClientCert) throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// create new channel with the provided name

ObjectName myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"lookupNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

if (myNewChannel != null)

throw new WLSAutomationException("NetworkChannel "+channelName+" already exists !");

// now create

myNewChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"createNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

// set protocol to http

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Protocol",new String("https")));

// set listener port

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ListenPort",new Integer(port)));

// enable

myJMXWrapper.setAttribute(myNewChannel,new Attribute("Enabled",new Boolean(true)));

// enable HttpEnabledForThisProtocol

myJMXWrapper.setAttribute(myNewChannel,new Attribute("HttpEnabledForThisProtocol",

new Boolean(true)));

myJMXWrapper.setAttribute(myNewChannel,new Attribute("TunnelingEnabled",new Boolean(false)));

// set OutboundEnabled

myJMXWrapper.setAttribute(myNewChannel,new Attribute("OutboundEnabled",new Boolean(false)));

// enable https

myJMXWrapper.setAttribute(myNewChannel,new Attribute("TwoWaySSLEnabled",new Boolean(true)));

myJMXWrapper.setAttribute(myNewChannel,new Attribute("ClientCertificateEnforced",

new Boolean(enforceClientCert)));

}

catch(Exception ex) {

throw new WLSAutomationException("Error while createHTTPSNetworkChannel ("+channelName+"): "+

ex.getMessage());

}

}

* add\_existing\_channel\_jmx

public void addNetworkChannel(String serverName, ObjectName myChannel)

throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// add the channel to this server

myJMXWrapper.invoke(myServerRuntime,

"addNetworkAccessPoint",

new Object[]{myChannel},

new String[]{ObjectName.class.getName()});

}

catch(Exception ex) {

throw new WLSAutomationException("Error while addNetworkChannel : "+ ex.getMessage());

}

}

* remove\_existing\_channel\_jmx

public void removeNetworkChannel(String serverName, ObjectName myChannel)

throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// remove the channel to this server

myJMXWrapper.invoke(myServerRuntime,

"removeNetworkAccessPoint",

new Object[]{myChannel},

new String[]{ObjectName.class.getName()});

}

catch(Exception ex) {

throw new WLSAutomationException("Error while addNetworkChannel : "+ ex.getMessage());

}

}

* setup\_virtual\_host\_jmx

public void createVirtualHost(String virtualhostName, // name

String channelName, // which networkchannel is attached

String[] targetServerNames, // targets

String[] networkNames // on which accept requests

) throws WLSAutomationException

{

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// lookup and try to find virtual host

ObjectName myNewVirtualHost = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupVirtualHost",

new Object[]{new String(virtualhostName)},

new String[]{String.class.getName()});

if (myNewVirtualHost != null)

throw new WLSAutomationException("Virtual host "+virtualhostName+" already exists !");

// create new virtual host with the provided name

myNewVirtualHost = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createVirtualHost",

new Object[]{new String(virtualhostName)},

new String[]{String.class.getName()});

// set referenced network channel

myJMXWrapper.setAttribute(myNewVirtualHost,new Attribute("NetworkAccessPoint",channelName));

// get servers and set it to virtual host targets

ObjectName[] myServerRuntimes = new ObjectName[targetServerNames.length];

for (int i=0;i<targetServerNames.length;i++)

myServerRuntimes[i] = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(targetServerNames[i])},

new String[]{String.class.getName()});

myJMXWrapper.setAttribute(myNewVirtualHost,new Attribute("Targets",myServerRuntimes));

// set VirtualHostNames

myJMXWrapper.setAttribute(myNewVirtualHost,new Attribute("VirtualHostNames",networkNames));

}

catch(Exception ex) {

throw new WLSAutomationException("Error while createVirtualHost ("+virtualhostName+"): "+ ex.getMessage());

}

}

* check\_session\_jmx

public boolean isEditSessionActive() throws WLSAutomationException

{

boolean sessionFound = false;

try

{

ObjectName myConfigurationManager = new ObjectName("com.bea:Name=ConfigurationManager,Type=weblogic.management.mbeanservers.edit.ConfigurationManagerMBean");

// try to get an active session. If call succeeds, then a session is available, otherwise an exception is thrown

CompositeData[] changes =

(CompositeData[])myJMXWrapper.getAttribute(myConfigurationManager, "UnactivatedChanges");

if (changes != null && changes.length>0)

{

// changes found

System.out.println("Active edit session found !");

// who is doing the changes

String userWhoEdits = (String)myJMXWrapper.getAttribute(myConfigurationManager,

"CurrentEditor");

System.out.println(" - The actual session is changed by "+ userWhoEdits +" !");

// number of changes waiting

System.out.println(" - The actual session has "+changes.length+" not activated changes ");

sessionFound = true;

}

else {

System.out.println("No active changes found !");

sessionFound = false;

}

return sessionFound;

}

catch(Exception ex)

{

System.out.println("Error while isEditSessionActive : "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_datasource\_jmx

/\*\*

\* Create a datasource either using user/password or using wallets

\*

\* @param datasourcename String Name of the new datasource

\* @param clustername String Name of the target cluster

\* @param jndiName String JNDI entry name

\* @param globalTransactionsProtocol String transaction setting

\* @param drivername String name of the driver

\* @param url String database URL - NOTE: in case of wallets don't forget the "/"

\* @param username String username - ONLY necessary if useWallet=false

\* @param password String password - ONLY necessary if useWallet=false

\* @param useWallet boolean does this datasource use wallets true/false

\* @param walletlocation String wallet directory - ONLY necessary if useWallet=true

\* @throws Exception

\*/

public void createDataSource(String datasourcename,

String clustername,

String jndiName,

String globalTransactionsProtocol,

String drivername,

String url,

String username,

String password,

boolean useWallet,

String walletlocation) throws Exception

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = new ObjectName("com.bea:Name=" + domainName +",Type=Domain");

// javax.management.ObjectName createJDBCSystemResource(name:java.lang.String )

ObjectName mySystemResourceMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createJDBCSystemResource",

new Object[]{new String(datasourcename)},

new String[]{String.class.getName()});

ObjectName myJDBCResourceMBean = (ObjectName)myJMXWrapper.getAttribute(mySystemResourceMBean,"JDBCResource");

ObjectName myJDBCDataSourceParamsMBean = (ObjectName)myJMXWrapper.getAttribute(myJDBCResourceMBean,"JDBCDataSourceParams");

ObjectName myJDBCDriverParamsMBean = (ObjectName)myJMXWrapper.getAttribute(myJDBCResourceMBean,"JDBCDriverParams");

ObjectName myJDBCConnectionPoolParamsMBean = (ObjectName)myJMXWrapper.getAttribute(myJDBCResourceMBean,"JDBCConnectionPoolParams");

// set attribute ListenAddress

myJMXWrapper.setAttribute(myJDBCResourceMBean,new Attribute("Name",datasourcename));

// set JNDI names

myJMXWrapper.setAttribute(myJDBCDataSourceParamsMBean,new Attribute("JNDINames",new String[]{jndiName}));

// set GlobalTransactionsProtocol

myJMXWrapper.setAttribute(myJDBCDataSourceParamsMBean,new Attribute("GlobalTransactionsProtocol",new String(globalTransactionsProtocol)));

// set TestTableName

myJMXWrapper.setAttribute(myJDBCConnectionPoolParamsMBean,new Attribute("TestTableName",new String("SQL SELECT \* FROM DUAL")));

// set URL

myJMXWrapper.setAttribute(myJDBCDriverParamsMBean,new Attribute("Url",new String(url)));

// set DriverName

myJMXWrapper.setAttribute(myJDBCDriverParamsMBean,

new Attribute("DriverName",new String(drivername)));

if (! useWallet)

{

// set Password

myJMXWrapper.setAttribute(myJDBCDriverParamsMBean,

new Attribute("Password", new String(password)));

// user

ObjectName myJDBCDriverParamsPropertiesMBean = (ObjectName)

myJMXWrapper.getAttribute(myJDBCDriverParamsMBean, "Properties");

ObjectName myUserPropertyMBean = null;

try {

myUserPropertyMBean = (ObjectName) myJMXWrapper.invoke(

myJDBCDriverParamsPropertiesMBean,

"lookupProperty",

new Object[] {new String("user")},

new String[] {String.class.getName()});

} catch (Exception ex) {

// ignore

}

if (myUserPropertyMBean == null)

myUserPropertyMBean = (ObjectName) myJMXWrapper.invoke(

myJDBCDriverParamsPropertiesMBean,

"createProperty",

new Object[] {new String("user")},

new String[] {String.class.getName()});

myJMXWrapper.setAttribute(myUserPropertyMBean, new Attribute("Value",

new String(username)));

}

else // WALLET !!

{

// user

ObjectName myJDBCDriverParamsPropertiesMBean = (ObjectName)

myJMXWrapper.getAttribute(myJDBCDriverParamsMBean, "Properties");

ObjectName myWalletLocationPropertyMBean = null;

try {

myWalletLocationPropertyMBean = (ObjectName) myJMXWrapper.invoke(

myJDBCDriverParamsPropertiesMBean,

"lookupProperty",

new Object[] {new String("oracle.net.wallet\_locatio")},

new String[] {String.class.getName()});

} catch (Exception ex) {

// ignore

}

if (myWalletLocationPropertyMBean == null)

myWalletLocationPropertyMBean = (ObjectName) myJMXWrapper.invoke(

myJDBCDriverParamsPropertiesMBean,

"createProperty",

new Object[] {new String("oracle.net.wallet\_locatio")},

new String[] {String.class.getName()});

myJMXWrapper.setAttribute(myWalletLocationPropertyMBean, new Attribute("Value",

new String(walletlocation)));

}

// Note that in this example it is exactly one cluster, but this does not need to be.

myJMXWrapper.setAttribute(mySystemResourceMBean,new Attribute("Targets",

new ObjectName[]{new ObjectName("com.bea:Name="+clustername+",Type=Cluster")} ));

}

* datasource\_MBeans

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/InternalProperties,Type=weblogic.j2ee.descriptor.wl.JDBCPropertiesBean

Attribute: Properties of Type : [Ljavax.management.ObjectName;

Operation: javax.management.ObjectName lookupProperty(java.lang.String:java.lang.String )

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: javax.management.ObjectName createProperty(java.lang.String:java.lang.String )

Operation: java.lang.Void destroyProperty(weblogic.j2ee.descriptor.wl.JDBCPropertyBean:javax.management.ObjectName )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Type=JDBCSystemResource

Attribute: Parent of Type : javax.management.ObjectName

Attribute: Resource of Type : javax.management.ObjectName

Attribute: Type of Type : java.lang.String

Attribute: CompatibilityName of Type : java.lang.String

Attribute: ModuleType of Type : java.lang.String

Attribute: SourcePath of Type : java.lang.String

Attribute: JDBCResource of Type : javax.management.ObjectName

Attribute: DescriptorFileName of Type : java.lang.String

Attribute: Notes of Type : java.lang.String

Attribute: Name of Type : java.lang.String

Attribute: SubDeployments of Type : [Ljavax.management.ObjectName;

Attribute: DeploymentPrincipalName of Type : java.lang.String

Attribute: Targets of Type : [Ljavax.management.ObjectName;

Attribute: DeploymentOrder of Type : java.lang.Integer

Operation: javax.management.ObjectName createSubDeployment(name:java.lang.String )

Operation: java.lang.Void destroySubDeployment(subDeployment:javax.management.ObjectName )

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: javax.management.ObjectName lookupSubDeployment(java.lang.String:java.lang.String )

Operation: java.lang.Void restoreDefaultValue(attributeName:java.lang.String )

Operation: java.lang.Void freezeCurrentValue(attributeName:java.lang.String )

Operation: java.lang.Void addTarget(target:javax.management.ObjectName )

Operation: java.lang.Void removeTarget(target:javax.management.ObjectName )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCDriverParams/Properties,Type=weblogic.j2ee.descriptor.wl.JDBCPropertiesBean

Attribute: Properties of Type : [Ljavax.management.ObjectName;

Operation: javax.management.ObjectName lookupProperty(java.lang.String:java.lang.String )

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: javax.management.ObjectName createProperty(java.lang.String:java.lang.String )

Operation: java.lang.Void destroyProperty(weblogic.j2ee.descriptor.wl.JDBCPropertyBean:javax.management.ObjectName )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

Operation: javax.management.ObjectName createProperty(java.lang.String:java.lang.String java.lang.String:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCDataSourceParams,Type=weblogic.j2ee.descriptor.wl.JDBCDataSourceParamsBean

Attribute: GlobalTransactionsProtocol of Type : java.lang.String

Attribute: RowPrefetch of Type : java.lang.Boolean

Attribute: RowPrefetchSize of Type : java.lang.Integer

Attribute: JNDINames of Type : [Ljava.lang.String;

Attribute: AlgorithmType of Type : java.lang.String

Attribute: ConnectionPoolFailoverCallbackHandler of Type : java.lang.String

Attribute: StreamChunkSize of Type : java.lang.Integer

Attribute: KeepConnAfterGlobalTx of Type : java.lang.Boolean

Attribute: KeepConnAfterLocalTx of Type : java.lang.Boolean

Attribute: FailoverRequestIfBusy of Type : java.lang.Boolean

Attribute: DataSourceList of Type : java.lang.String

Attribute: Scope of Type : java.lang.String

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: java.lang.Void addJNDIName(java.lang.String:java.lang.String )

Operation: java.lang.Void removeJNDIName(java.lang.String:java.lang.String )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCConnectionPoolParams,Type=weblogic.j2ee.descriptor.wl.JDBCConnectionPoolParamsBean

Attribute: ProfileType of Type : java.lang.Integer

Attribute: ConnectionHarvestTriggerCount of Type : java.lang.Integer

Attribute: StatementCacheSize of Type : java.lang.Integer

Attribute: TestConnectionsOnReserve of Type : java.lang.Boolean

Attribute: RemoveInfectedConnections of Type : java.lang.Boolean

Attribute: WrapTypes of Type : java.lang.Boolean

Attribute: FatalErrorCodes of Type : java.lang.String

Attribute: InactiveConnectionTimeoutSeconds of Type : java.lang.Integer

Attribute: TestTableName of Type : java.lang.String

Attribute: LoginDelaySeconds of Type : java.lang.Integer

Attribute: PinnedToThread of Type : java.lang.Boolean

Attribute: ConnectionHarvestMaxCount of Type : java.lang.Integer

Attribute: IgnoreInUseConnectionsEnabled of Type : java.lang.Boolean

Attribute: SecondsToTrustAnIdlePoolConnection of Type : java.lang.Integer

Attribute: DriverInterceptor of Type : java.lang.String

Attribute: ConnectionReserveTimeoutSeconds of Type : java.lang.Integer

Attribute: JDBCXADebugLevel of Type : java.lang.Integer

Attribute: TestFrequencySeconds of Type : java.lang.Integer

Attribute: ShrinkFrequencySeconds of Type : java.lang.Integer

Attribute: StatementTimeout of Type : java.lang.Integer

Attribute: MaxCapacity of Type : java.lang.Integer

Attribute: MinCapacity of Type : java.lang.Integer

Attribute: CredentialMappingEnabled of Type : java.lang.Boolean

Attribute: ConnectionLabelingCallback of Type : java.lang.String

Attribute: HighestNumWaiters of Type : java.lang.Integer

Attribute: IdentityBasedConnectionPoolingEnabled of Type : java.lang.Boolean

Attribute: InitSql of Type : java.lang.String

Attribute: InitialCapacity of Type : java.lang.Integer

Attribute: StatementCacheType of Type : java.lang.String

Attribute: ConnectionCreationRetryFrequencySeconds of Type : java.lang.Integer

Attribute: CapacityIncrement of Type : java.lang.Integer

Attribute: ProfileHarvestFrequencySeconds of Type : java.lang.Integer

Operation: java.lang.Void unSet(propertyName:java.lang.String )

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource,Type=weblogic.j2ee.descriptor.wl.JDBCDataSourceBean

Attribute: JDBCDataSourceParams of Type : javax.management.ObjectName

Attribute: JDBCDriverParams of Type : javax.management.ObjectName

Attribute: InternalProperties of Type : javax.management.ObjectName

Attribute: Name of Type : java.lang.String

Attribute: JDBCXAParams of Type : javax.management.ObjectName

Attribute: JDBCConnectionPoolParams of Type : javax.management.ObjectName

Attribute: JDBCOracleParams of Type : javax.management.ObjectName

Attribute: Version of Type : java.lang.String

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCDriverParams,Type=weblogic.j2ee.descriptor.wl.JDBCDriverParamsBean

Attribute: PasswordEncrypted of Type : [B

Attribute: UsePasswordIndirection of Type : java.lang.Boolean

Attribute: Password of Type : java.lang.String

Attribute: DriverName of Type : java.lang.String

Attribute: Properties of Type : javax.management.ObjectName

Attribute: UseXaDataSourceInterface of Type : java.lang.Boolean

Attribute: Url of Type : java.lang.String

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=AdminServer,Server=AdminServer,Type=WLDFServerDiagnostic

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCOracleParams,Type=weblogic.j2ee.descriptor.wl.JDBCOracleParamsBean

Attribute: OnsWalletFile of Type : java.lang.String

Attribute: ConnectionInitializationCallback of Type : java.lang.String

Attribute: FanEnabled of Type : java.lang.Boolean

Attribute: OnsNodeList of Type : java.lang.String

Attribute: UseDatabaseCredentials of Type : java.lang.Boolean

Attribute: OracleProxySession of Type : java.lang.Boolean

Attribute: OracleEnableJavaNetFastPath of Type : java.lang.Boolean

Attribute: OnsWalletPasswordEncrypted of Type : [B

Attribute: OracleOptimizeUtf8Conversion of Type : java.lang.Boolean

Attribute: OnsWalletPassword of Type : java.lang.String

Attribute: AffinityPolicy of Type : java.lang.String

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=user,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCDriverParams/Properties/Properties[user],Type=weblogic.j2ee.descriptor.wl.JDBCPropertyBean

Attribute: Value of Type : java.lang.String

Attribute: SysPropValue of Type : java.lang.String

Attribute: Name of Type : java.lang.String

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

Operation: java.lang.Void unSet(propertyName:java.lang.String )

com.bea:Name=MySuperDataSource,Parent=[TestDomain]/JDBCSystemResources[MySuperDataSource],Path=JDBCResource[MySuperDataSource]/JDBCXAParams,Type=weblogic.j2ee.descriptor.wl.JDBCXAParamsBean

Attribute: RecoverOnlyOnce of Type : java.lang.Boolean

Attribute: KeepLogicalConnOpenOnRelease of Type : java.lang.Boolean

Attribute: ResourceHealthMonitoring of Type : java.lang.Boolean

Attribute: NewXaConnForCommit of Type : java.lang.Boolean

Attribute: XaRetryDurationSeconds of Type : java.lang.Integer

Attribute: XaEndOnlyOnce of Type : java.lang.Boolean

Attribute: RollbackLocalTxUponConnClose of Type : java.lang.Boolean

Attribute: KeepXaConnTillTxComplete of Type : java.lang.Boolean

Attribute: XaTransactionTimeout of Type : java.lang.Integer

Attribute: XaRetryIntervalSeconds of Type : java.lang.Integer

Attribute: XaSetTransactionTimeout of Type : java.lang.Boolean

Attribute: NeedTxCtxOnClose of Type : java.lang.Boolean

Operation: java.lang.Boolean isSet(propertyName:java.lang.String )

* create\_filestore\_jmx

public ObjectName createFileStore(String fileStoreName, String wlsServerName) throws WLSAutomationException

{

try

{

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// create filesstore and configure it

ObjectName newFileStore = (ObjectName)myJMXWrapper.invoke(

domainRoot,"createFileStore",

new Object[]{fileStoreName},new String[]{String.class.getName()});

myJMXWrapper.setAttribute(newFileStore,

new Attribute("Directory",

"/weblogic\_domains/MartinTest\_Domain"));

// define targets

myJMXWrapper.setAttribute(newFileStore, new Attribute("Targets",

new ObjectName[]{new ObjectName("com.bea:Name="+wlsServerName+",

Type=Server")}));

return newFileStore;

}

catch(Exception ex)

{

System.out.println("Error while createFileStore ("+fileStoreName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* JMS\_server\_instance\_jmx

public void createAnewJMSServer(String jmsServerName, String fileStoreName, String wlsServerName)

throws WLSAutomationException

{

try {

// 1st create the filestore:

ObjectName newFileStore = createFileStore(fileStoreName, wlsServerName);

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// now create the server

ObjectName newJMSServer = (ObjectName)myJMXWrapper.invoke(domainRoot,"createJMSServer",

new Object[]{jmsServerName},new String[]{String.class.getName()});

// set the persistent store

myJMXWrapper.setAttribute(newJMSServer, new Attribute("PersistentStore", newFileStore));

myJMXWrapper.setAttribute(newJMSServer,

new Attribute("Targets", new ObjectName[]{

new ObjectName("com.bea:Name="+wlsServerName+",Type=Server")}));

// now we can define threshold values

myJMXWrapper.setAttribute(newJMSServer, new Attribute("BytesThresholdHigh", new Long(-1)));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("BytesThresholdLow", new Long(-1)));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("MessagesThresholdHigh",

new Long(-1)));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("MessagesThresholdLow",

new Long(-1)));

// now we can define quotas values

myJMXWrapper.setAttribute(newJMSServer, new Attribute("BytesMaximum", new Long(-1)));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("MessagesMaximum", new Long(-1)));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("BlockingSendPolicy", "FIFO"));

myJMXWrapper.setAttribute(newJMSServer, new Attribute("MaximumMessageSize",

new Integer(10000000)));

// now configure the log file

// similiar to com.bea:Name=MyTestServer\_1,Type=JMSMessageLogFile,JMSServer=MyTestServer\_1

ObjectName newJMSServerLogFile = (ObjectName)myJMXWrapper.getAttribute(newJMSServer,

"JMSMessageLogFile");

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("RotationType",

new String("byTime")));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("RotateLogOnStartup",

new Boolean(false)));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("RotationTime",

new String("00:00")));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("FileTimeSpan",

new Integer(24)));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("FileCount", new Integer(25)));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("NumberOfFilesLimited",

new Boolean(true)));

myJMXWrapper.setAttribute(newJMSServerLogFile, new Attribute("FileName",

new String(jmsServerName+".log")));

}

catch(Exception ex)

{

System.out.println("Error while createAnewJMSServer ("+jmsServerName+"): "+

ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_JMS\_modules\_jmx

public void createJMSModule(String jmsModuleName, String targetType, String targetName) throws WLSAutomationException

{

try

{

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// now create the server

ObjectName newJMSModule = (ObjectName)myJMXWrapper.invoke(domainRoot,

"createJMSSystemResource",

new Object[]{jmsModuleName},

new String[]{String.class.getName()});

// set the target

myJMXWrapper.setAttribute(newJMSModule, new Attribute("Targets",

new ObjectName[]{new ObjectName("com.bea:Name="+targetName+",

Type="+targetType)}));

}

catch(Exception ex)

{

System.out.println("Error while createJMSModule ("+jmsModuleName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_connection\_factory\_jmx

public void createJmsConnectionFactory(String jmsModuleName, String connectionFactoryName,

String jmsJNDIname) throws WLSAutomationException

{

try {

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",

Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,

"JMSResource");

// create the connection factory

ObjectName myJMSModuleConnfactory =

(ObjectName)(ObjectName)myJMXWrapper.invoke(myJMSModuleResource,

"createConnectionFactory",

new Object[]{connectionFactoryName},

new String[]{String.class.getName()});

// configure connection factory

myJMXWrapper.setAttribute(myJMSModuleConnfactory, new Attribute("JNDIName",

new String(jmsJNDIname)));

myJMXWrapper.setAttribute(myJMSModuleConnfactory, new Attribute("DefaultTargetingEnabled",

new Boolean(true)));

// get the security params

ObjectName mySecParams = (ObjectName)myJMXWrapper.getAttribute(myJMSModuleConnfactory,

"SecurityParams");

// set AttachJMSXUserId

myJMXWrapper.setAttribute(mySecParams, new Attribute("AttachJMSXUserId", new Boolean(false)));

// get the default delivery params

ObjectName myDelivParams = (ObjectName)myJMXWrapper.getAttribute(myJMSModuleConnfactory,

"DefaultDeliveryParams");

// set AttachJMSXUserId

myJMXWrapper.setAttribute(myDelivParams, new Attribute("DefaultDeliveryMode",

new String("Persistent")));

myJMXWrapper.setAttribute(myDelivParams, new Attribute("DefaultTimeToLive", new Integer(0)));

myJMXWrapper.setAttribute(myDelivParams, new Attribute("DefaultPriority", new Integer(2)));

}

catch(Exception ex) {

System.out.println("Error while createJmsConnectionFactory ("+jmsModuleName+":"+

connectionFactoryName+"): " + ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* XA\_enabled\_connection\_factory\_jmx

public void createJms\_XA\_ConnectionFactory(String jmsModuleName, String connectionFactoryName,

String jmsJNDIname)

throws WLSAutomationException

{

try {

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",

Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,

"JMSResource");

// create the connection factory

ObjectName myJMSModuleConnfactory=

(ObjectName)(ObjectName)myJMXWrapper.invoke(myJMSModuleResource,

"createConnectionFactory",

new Object[]{connectionFactoryName},

new String[]{String.class.getName()});

// configure connection factory

myJMXWrapper.setAttribute(myJMSModuleConnfactory, new Attribute("JNDIName",

new String(jmsJNDIname)));

myJMXWrapper.setAttribute(myJMSModuleConnfactory, new Attribute("DefaultTargetingEnabled",

new Boolean(true)));

// get the security params

ObjectName mySecParams = (ObjectName)myJMXWrapper.getAttribute(myJMSModuleConnfactory,

"SecurityParams");

// set AttachJMSXUserId

myJMXWrapper.setAttribute(mySecParams, new Attribute("AttachJMSXUserId",

new Boolean(false)));

// get the transaction params

ObjectName myTxParams = (ObjectName)myJMXWrapper.getAttribute(myJMSModuleConnfactory,

"TransactionParams");

// configure it

myJMXWrapper.setAttribute(myTxParams, new Attribute("TransactionTimeout",

new Integer(3600)));

myJMXWrapper.setAttribute(myTxParams, new Attribute("XAConnectionFactoryEnabled",

new Boolean(true)));

}

catch(Exception ex)

{

System.out.println("Error while createJms\_XA\_ConnectionFactory ("+jmsModuleName+":"+

connectionFactoryName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* subdeployment\_jmx

public void createJMSSubDeployment(String jmsModuleName, String subDeploymentName, String targetName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",

Type=JMSSystemResource");

// get the resource

//ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,

"JMSResource");

// create subdeployment

ObjectName myJMSModuleSubDeploy =

(ObjectName)myJMXWrapper.invoke(myJMSModule,"createSubDeployment",

new Object[]{subDeploymentName},

new String[]{String.class.getName()});

// target the new subdeployment

// set the target

myJMXWrapper.setAttribute(myJMSModuleSubDeploy, new Attribute("Targets",

new ObjectName[]{new ObjectName("com.bea:Name="+targetName+",Type=JMSServer")}));

}

catch(Exception ex)

{

System.out.println("Error while createJMSSubDeployment ("+jmsModuleName+":"+

subDeploymentName+"): " + ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* queue\_jmx

// Creating queue

public void createQueue(String jmsModuleName, String queueName, String jndiQueueName,

String subDeploymentName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",

Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,

"JMSResource");

// create queue

ObjectName myJMSQueue = (ObjectName)(ObjectName)myJMXWrapper.invoke(myJMSModuleResource,

"createQueue",new Object[]{queueName},new String[]{String.class.getName()});

// configue queue

myJMXWrapper.setAttribute(myJMSQueue, new Attribute("JNDIName",

new String(jndiQueueName)));

myJMXWrapper.setAttribute(myJMSQueue, new Attribute("SubDeploymentName",

new String(subDeploymentName)));

}

catch(Exception ex)

{

System.out.println("Error while createQueue ("+jmsModuleName+":"+queueName+"): "+

ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* queue\_error\_handling\_jmx

public void createQueueWithErrorHandling(String jmsModuleName, String queueName, String jndiQueueName, String subDeploymentName, String errorQueueName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",

Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,"JMSResource");

// create queue

ObjectName myJMSQueue = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"createQueue",

new Object[]{queueName},new String[]{String.class.getName()});

// configure queue

myJMXWrapper.setAttribute(myJMSQueue, new Attribute("JNDIName", new String(jndiQueueName)));

myJMXWrapper.setAttribute(myJMSQueue, new Attribute("SubDeploymentName", new String(subDeploymentName)));

// get the default delivery params

ObjectName myFailureParams = (ObjectName)myJMXWrapper.getAttribute(myJMSQueue,"DeliveryFailureParams");

// set AttachJMSXUserId

myJMXWrapper.setAttribute(myFailureParams, new Attribute("RedeliveryLimit", new Integer(3)));

myJMXWrapper.setAttribute(myFailureParams, new Attribute("ExpirationPolicy", new String("Redirect")));

// lookup error queue on SAME module (can be done more generic if error queue is located on a different module)

ObjectName errorQueue = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"lookupQueue",

new Object[]{errorQueueName},new String[]{String.class.getName()});

// set error queue

myJMXWrapper.setAttribute(myQueueDeliveryFailureParams, new Attribute("ErrorDestination",errorQueue));

}

catch(Exception ex)

{

System.out.println("Error while createQueueWithErrorHandling ("+jmsModuleName+":"+queueName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_topic\_jmx

public void createTopic(String jmsModuleName, String topicName, String jndiTopicName, String subDeploymentName)

throws WLSAutomationException

{

try {

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,"JMSResource");

// create topic

ObjectName myJMSTopic = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"createTopic",

new Object[]{topicName},new String[]{String.class.getName()});

// configure topic

myJMXWrapper.setAttribute(myJMSTopic, new Attribute("JNDIName", new String(jndiTopicName)));

myJMXWrapper.setAttribute(myJMSTopic, new Attribute("SubDeploymentName", new String(subDeploymentName)));

}

catch(Exception ex)

{

System.out.println("Error while createTopic ("+jmsModuleName+":"+topicName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* JNDI\_provider\_jmx

public void createForeignJNDIProvider(String providerName, ObjectName[] targets, java.util.Properties properties)

throws WLSAutomationException {

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

ObjectName myProviderMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupForeignJNDIProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myProviderMBean==null) {

// create

// Operation: javax.management.ObjectName createForeignJNDIProvider(name:java.lang.String )

myProviderMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createForeignJNDIProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

//target to cluster

myJMXWrapper.setAttribute(myProviderMBean, new Attribute("Targets",targets));

// configure

if (properties.containsKey("INITIALCONTEXTFACTORY"))

myJMXWrapper.setAttribute(myProviderMBean,

new Attribute("InitialContextFactory",properties.get("INITIALCONTEXTFACTORY")));

if (properties.containsKey("PROVIDERURL"))

myJMXWrapper.setAttribute(myProviderMBean, new Attribute("ProviderURL",properties.get("PROVIDERURL")));

if (properties.containsKey("USER"))

myJMXWrapper.setAttribute(myProviderMBean, new Attribute("User",properties.get("USER")));

if (properties.containsKey("PASSWORD"))

myJMXWrapper.setAttribute(myProviderMBean, new Attribute("Password",properties.get("PASSWORD")));

// create and configure JNDI link

// Operation: javax.management.ObjectName createForeignJNDILink(name:java.lang.String )

ObjectName myForeignLinkMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createForeignJNDILink",

new Object[]{new String(providerName+"\_Link")},

new String[]{String.class.getName()});

if (properties.containsKey("LOCALJNDINAME"))

myJMXWrapper.setAttribute(myForeignLinkMBean,

new Attribute("LocalJNDIName",properties.get("LOCALJNDINAME")));

if (properties.containsKey("REMOTEJNDINAME"))

myJMXWrapper.setAttribute(myForeignLinkMBean,

new Attribute("RemoteJNDIName",properties.get("REMOTEJNDINAME")));

}

else

throw new WLSAutomationException("Foreign JNDI provider "+providerName+" already exist - cannot create !");

}

catch(Exception ex)

{

throw new WLSAutomationException(ex);

}

}

* setup\_mail\_session\_jmx

public void createMailSession(String mailSessionName, String jndiName,

ObjectName[] targets,Properties properties) throws WLSAutomationExceptio {

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myMailSessionMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupMailSession",

new Object[]{new String(mailSessionName)},

new String[]{String.class.getName()});

if (myMailSessionMBean==null)

{

// create

// Operation: javax.management.ObjectName createMailSession(name:java.lang.String )

myMailSessionMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"createMailSession",

new Object[]{new String(mailSessionName)},

new String[]{String.class.getName()});

//target to targets

// e.g. set('Targets',jarray.array([ObjectName('com.bea:Name=MartinTest\_Cluster,Type=Cluster')], ObjectName))

myJMXWrapper.setAttribute(myMailSessionMBean, new Attribute("Targets",targets));

// configure

myJMXWrapper.setAttribute(myMailSessionMBean, new Attribute("JNDIName",jndiName));

myJMXWrapper.setAttribute(myMailSessionMBean, new Attribute("Properties",properties));

}

else

throw new WLSAutomationException("Mail session with name "+mailSessionName+" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* minimum\_thread\_constraint\_jmx

public ObjectName createMinThreadsConstraint(String newName, int count, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// check if already created

ObjectName myMinThreadsConstraint = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupMinThreadsConstraint",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myMinThreadsConstraint==null){

// create

myMinThreadsConstraint = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createMinThreadsConstraint",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// configure

myJMXWrapper.setAttribute(myMinThreadsConstraint,

new Attribute("Count",new Integer(count)));

if (targets != null)

myJMXWrapper.setAttribute(myMinThreadsConstraint, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("MinThreadsConstraint with name "+newName+

" already exist - cannot create !");

return myMinThreadsConstraint;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* maximum\_thread\_constraint\_jmx

public ObjectName createMaxThreadsConstraint(String newName, int count, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// check if already created

ObjectName myMaxThreadsConstraint = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupMaxThreadsConstraint",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myMaxThreadsConstraint==null){

// create

myMaxThreadsConstraint = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createMaxThreadsConstraint",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// configure

myJMXWrapper.setAttribute(myMaxThreadsConstraint,

new Attribute("Count",new Integer(count)));

if (targets != null)

myJMXWrapper.setAttribute(myMaxThreadsConstraint, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("MaxThreadsConstraint with name "+newName+

" already exist - cannot create !");

return myMaxThreadsConstraint;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* capacity\_constraint\_jmx

public ObjectName createCapacity(String newName, int count, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// check if already created

ObjectName myCapacity = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupCapacity",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myCapacity==null){

// create

myCapacity = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createCapacity",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// configure

myJMXWrapper.setAttribute(myCapacity,

new Attribute("Count",new Integer(count)));

if (targets != null)

myJMXWrapper.setAttribute(myCapacity, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("Capacity with name "+newName+

" already exist - cannot create !");

return myCapacity;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* fair\_share\_jmx

public ObjectName createFairShareRequestClass(String newName, int count, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// check if already created

ObjectName myFairShareRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupFairShareRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myFairShareRequestClass==null){

// create

myFairShareRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createFairShareRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// configure

myJMXWrapper.setAttribute(myFairShareRequestClass,

new Attribute("Count",new Integer(count)));

if (targets != null)

myJMXWrapper.setAttribute(myFairShareRequestClass, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("FairShareRequestClass with name "+newName+

" already exist - cannot create !");

return myFairShareRequestClass;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* response\_time\_jmx

public ObjectName createResponseTimeRequestClass(String newName, int count, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// check if already created

ObjectName myResponseTimeRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupResponseTimeRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myResponseTimeRequestClass==null){

// create

myResponseTimeRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createResponseTimeRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// configure

myJMXWrapper.setAttribute(myResponseTimeRequestClass,

new Attribute("Count",new Integer(count)));

if (targets != null)

myJMXWrapper.setAttribute(myResponseTimeRequestClass, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("ResponseTimeRequestClass with name "+newName+

" already exist - cannot create !");

return myResponseTimeRequestClass;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* context\_jmx

public ObjectName createContextRequestClass(String newName,

ArrayList<ContextCaseData> myContextList, ObjectName[] targets)

throws WLSAutomationException {

try

{ // lookup selftuning

ObjectName mySelfTuning = lookupSelfTuning();

// Operation: javax.management.ObjectName lookupMinThreadsConstraint(name:java.lang.String )

ObjectName myContextRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"lookupContextRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

if (myContextRequestClass==null) {

// create

myContextRequestClass = (ObjectName)myJMXWrapper.invoke(mySelfTuning,

"createContextRequestClass",

new Object[]{new String(newName)}, new String[]{String.class.getName()});

// create all the context sub-MBeans

for (int i=0;i<myContextList.size();i++) {

ContextCaseData myData = myContextList.get(i);

ObjectName newContext = (ObjectName)myJMXWrapper.invoke(myContextRequestClass,

"createContextCase",

new Object[]{new String("Context\_"+i)},

new String[]{String.class.getName()});

myJMXWrapper.setAttribute(newContext, new Attribute("UserName", myData.getUserName()));

myJMXWrapper.setAttribute(newContext, new Attribute("GroupName", myData.getGroupName()));

myJMXWrapper.setAttribute(newContext,

new Attribute("RequestClassName", myData.getRequestClassName()));

if (targets != null)

myJMXWrapper.setAttribute(newContext, new Attribute("Targets",targets));

}

if (targets != null)

myJMXWrapper.setAttribute(myContextRequestClass, new Attribute("Targets",targets));

}

else

throw new WLSAutomationException("ContextRequestClass with name "+newName+" already exist !”);

return myContextRequestClass;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* resource\_deletion

destroyRemoteSAFContext(remoteSAFContext:javax.management.ObjectName )

destroyWTCServer(wtcServer:javax.management.ObjectName )

destroyApplication(application:javax.management.ObjectName )

destroyXMLEntityCache(XMLEntityCache:javax.management.ObjectName )

destroySNMPAgentDeployment(mbean:javax.management.ObjectName )

destroyVirtualHost(host:javax.management.ObjectName )

destroyRealm(weblogic.management.configuration.RealmMBean:javax.management.ObjectName )

destroyFileRealm(weblogic.management.configuration.FileRealmMBean:javax.management.ObjectName )

destroyForeignJMSDestination(wsc:javax.management.ObjectName )

destroyJMSBridgeDestination(jmsBridgeDestination:javax.management.ObjectName )

destroyJMSInteropModule(bean:javax.management.ObjectName )

destroyJoltConnectionPool(joltConnectionPool:javax.management.ObjectName )

destroyLogFilter(logFilter:javax.management.ObjectName )

destroyCluster(cluster:javax.management.ObjectName )

destroyJMSConnectionConsumer(wsc:javax.management.ObjectName )

destroyJDBCSystemResource(bean:javax.management.ObjectName )

destroySAFAgent(sAFAgent:javax.management.ObjectName )

destroyLDAPRealm(weblogic.management.configuration.LDAPRealmMBean:javax.management.ObjectName )

destroyErrorHandling(errorHandling:javax.management.ObjectName )

destroyBridgeDestination(bridgeDestination:javax.management.ObjectName )

destroyPathService(pathService:javax.management.ObjectName )

destroyMailSession(ms:javax.management.ObjectName )

destroyMigratableTarget(bean:javax.management.ObjectName )

destroyWLDFSystemResource(bean:javax.management.ObjectName )

destroyJDBCStore(store:javax.management.ObjectName )

destroyJMSServer(jmsServer:javax.management.ObjectName )

destroyWSReliableDeliveryPolicy(policy:javax.management.ObjectName )

destroySingletonService(sc:javax.management.ObjectName )

destroyUnixRealm(weblogic.management.configuration.UnixRealmMBean:javax.management.ObjectName )

destroyStartupClass(sc:javax.management.ObjectName )

destroyEJBContainer()

destroyServer(server:javax.management.ObjectName )

destroyForeignJNDIProvider(provider:javax.management.ObjectName )

destroyWebserviceSecurity(wsc:javax.management.ObjectName )

destroyCustomResource(bean:javax.management.ObjectName )

destroyMessagingBridge(bridge:javax.management.ObjectName )

destroyCachingRealm(weblogic.management.configuration.CachingRealmMBean:javax.management.ObjectName )

destroyMachine(machine:javax.management.ObjectName )

destroyJMSSystemResource(bean:javax.management.ObjectName )

destroyDomainLogFilter(logFilter:javax.management.ObjectName )

destroyNTRealm(weblogic.management.configuration.NTRealmMBean:javax.management.ObjectName )

destroyShutdownClass(sc:javax.management.ObjectName )

destroyCoherenceClusterSystemResource(bean:javax.management.ObjectName )

destroyMigratableRMIService(bean:javax.management.ObjectName )

destroyFileStore(store:javax.management.ObjectName )

destroyXMLRegistry(registry:javax.management.ObjectName )

destroyCoherenceServer(bean:javax.management.ObjectName )

destroyPasswordPolicy(weblogic.management.configuration.PasswordPolicyMBean:javax.management.ObjectName )

destroyFileT3(fileT3:javax.management.ObjectName )

destroyRDBMSRealm(weblogic.management.configuration.RDBMSRealmMBean:javax.management.ObjectName )

destroyWLECConnectionPool(store:javax.management.ObjectName )

* delete\_datasource\_jmx

public void deleteDataSource(String datasourcename) throws Exception

{

try

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupJDBCSystemResource(name:java.lang.String )

ObjectName mySystemResourceMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupJDBCSystemResource",

new Object[]{new String(datasourcename)},

new String[]{String.class.getName()});

if (mySystemResourceMBean!=null) {

// ok, found => delete it now !

myJMXWrapper.invoke(myDomainMBean,"destroyJDBCSystemResource",

new Object[]{mySystemResourceMBean},new String[]{ObjectName.class.getName()});

}

else

throw new WLSAutomationException("Datasource "+datasourcename+" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_filestore\_jmx

public void destroyFileStore(String fileStoreName) throws WLSAutomationException

{

try {

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// lookup filestore as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myFileStore = (ObjectName)myJMXWrapper.invoke(domainRoot,"lookupFileStore",new Object[]{fileStoreName},

new String[]{String.class.getName()});

// destroy filesstore

myJMXWrapper.invoke(domainRoot,"destroyFileStore",new Object[]{myFileStore},new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyFileStore ("+fileStoreName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* destroy\_JMS\_server\_jmx

public void destroyJMSServer(String jmsServerName) throws WLSAutomationException

{

try {

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// lookup jmsserver as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myJMSServer = (ObjectName)myJMXWrapper.invoke(domainRoot,"lookupJMSServer",

new Object[]{jmsServerName},new String[]{String.class.getName()});

// destroy filesstore

myJMXWrapper.invoke(domainRoot,"destroyJMSServer",new Object[]{myJMSServer},new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyJMSServer ("+jmsServerName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* delete\_JMS\_module\_jmx

public void destroyJMSModule(String jmsModuleName) throws WLSAutomationException

{

try {

// get the domain config root

ObjectName domainRoot = myJMXWrapper.getDomainConfigRoot();

// lookup jmsmodule as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myJMSModule = (ObjectName)myJMXWrapper.invoke(domainRoot,"lookupJMSSystemResource",

new Object[]{jmsModuleName},new String[]{String.class.getName()});

// destroy filesstore

myJMXWrapper.invoke(domainRoot,"destroyJMSSystemResource",new Object[]{myJMSModule},

new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyJMSModule ("+jmsModuleName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* destroy\_connection\_factory\_jmx

public void destroyJmsConnectionFactory(String jmsModuleName, String connectionFactoryName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,"JMSResource");

// lookup connection factory from module as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myJMSConnFactory = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"lookupConnectionFactory",

new Object[]{connectionFactoryName},new String[]{String.class.getName()});

// destroy the connection factory

myJMXWrapper.invoke(myJMSModuleResource,"destroyConnectionFactory",new Object[]{myJMSConnFactory},

new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroying JmsConnectionFactory ("+jmsModuleName+":"+connectionFactoryName+"): "

+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* destroy\_subdeployment\_jmx

public void destroyJMSSubDeployment(String jmsModuleName, String subDeploymentName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",Type=JMSSystemResource");

// lookup sub deplyoment from module as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myJMSSubDeployment = (ObjectName)myJMXWrapper.invoke(myJMSModule,"lookupSubDeployment",new Object[]{subDeploymentName},new String[]{String.class.getName()});

// destroy the connection factory

myJMXWrapper.invoke(myJMSModule,"destroySubDeployment",new Object[]{myJMSSubDeployment},new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyJMSSubDeployment ("+jmsModuleName+":"+subDeploymentName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* destroy\_queue\_jmx

public void destroyQueue(String jmsModuleName, String queueName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,"JMSResource");

// lookup queue from moduleResource as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myQueue = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"lookupQueue",new Object[]{queueName},new String[]{String.class.getName()});

// destroy the queue

myJMXWrapper.invoke(myJMSModuleResource,"destroyQueue",new Object[]{myQueue},new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyQueue ("+jmsModuleName+":"+queueName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* destroy\_topic\_jmx

public void destroyTopic(String jmsModuleName, String topicName) throws WLSAutomationException

{

try

{

// get the jmsModuleName config root

ObjectName myJMSModule = new ObjectName("com.bea:Name="+jmsModuleName+",Type=JMSSystemResource");

// get the resource

ObjectName myJMSModuleResource = (ObjectName)myJMXWrapper.getAttribute(myJMSModule,"JMSResource");

// lookup topic from moduleResource as ObjectName is needed (alternatively you can construct the ObjectName)

ObjectName myTopic = (ObjectName)myJMXWrapper.invoke(myJMSModuleResource,"lookupTopic",new Object[]{topicName},new String[]{String.class.getName()});

// destroy the topic

myJMXWrapper.invoke(myJMSModuleResource,"destroyTopic",new Object[]{myTopic},new String[]{ObjectName.class.getName()});

}

catch(Exception ex)

{

System.out.println("Error while destroyTopic ("+jmsModuleName+":"+topicName+"): "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* delete\_JNDI\_provider\_jmx

public void deleteForeignJNDIProvider(String providerName) throws WLSAutomationException

{

try

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

ObjectName myProviderMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupForeignJNDIProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myProviderMBean!=null)

{

// delete

// Operation: java.lang.Void destroyForeignJNDIProvider(provider:javax.management.ObjectName )

myJMXWrapper.invoke(myDomainMBean,"destroyForeignJNDIProvider",

new Object[]{new String(providerName)},new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("Foreign JNDI provider "+providerName+" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_mail\_session\_jmx

public void deleteMailSession(String mailSessionName) throws WLSAutomationException {

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

ObjectName myMailSessionMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupMailSession",

new Object[]{new String(mailSessionName)},

new String[]{String.class.getName()});

if (myMailSessionMBean!=null)

{

// delete

// Operation: java.lang.Void destroyMailSession(ms:javax.management.ObjectName )

myJMXWrapper.invoke(myDomainMBean,"destroyMailSession",new Object[]{myMailSessionMBean},

new String[]{ObjectName.class.getName()});

}

else

throw new WLSAutomationException("Mail session with name "+mailSessionName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_managed-server\_jmx

public void deleteManagedServer(String managedServerName, boolean deleteAlsoIfDependenciesExist) throws WLSAutomationException

{

try

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot(); // new ObjectName("com.bea:Name=" + domainName +",Type=Domain");

// Operation: javax.management.ObjectName lookupServer(name:java.lang.String )

ObjectName myServer = (ObjectName)myJMXWrapper.invoke(myDomainMBean,"lookupServer",new Object[]{managedServerName},new String[]{String.class.getName()});

if (myServer == null)

throw new WLSAutomationException("Server "+managedServerName+" does not exist !");

else

{

// server exists

if (! deleteAlsoIfDependenciesExist)

{

// check for dependencies

if (managedserverHostsApplications(managedServerName))

throw new WLSAutomationException("Applications still deployed on server "+managedServerName+" - cannot delete !");

if (managedserverHostsDatasources(managedServerName))

throw new WLSAutomationException("Datasources still deployed on server "+managedServerName+" - cannot delete !");

}

// ok, can delete

System.out.println("Managed Server "+managedServerName+" will be destroyed !");

// detach from cluster if any

myJMXWrapper.setAttribute(myServer, new Attribute("Cluster", null));

// detach from machine if any

myJMXWrapper.setAttribute(myServer, new Attribute("Machine", null));

// destroy server

myJMXWrapper.invoke(myDomainMBean,"destroyServer",new Object[]{myServer},new String[]{ObjectName.class.getName()});

}

}

catch(Exception ex) {

throw new WLSAutomationException("Error while deleteManagedServer ("+managedServerName+"): "+ ex.getMessage());

}

}

* delete\_cluster\_jmx

public void deleteCluster(String clusterName, boolean deleteAlsoIfDependenciesExist) throws WLSAutomationException

{

try

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

ObjectName myCluster = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupCluster",new Object[]{clusterName},new String[]{String.class.getName()});

if (myCluster == null)

throw new WLSAutomationException("Cluster "+clusterName+" does not exist !");

else

{

// cluster exists

if (! deleteAlsoIfDependenciesExist)

{

// check for dependencies

if (clusterHasManagedServers(clusterName))

throw new WLSAutomationException("Cluster "+clusterName+" still has server members - cannot delete !");

}

// ok delete cluster

ObjectName[] clusterMembers = (ObjectName[])myJMXWrapper.getAttribute(myCluster,"Servers");

if (clusterMembers!=null && clusterMembers.length>0)

{

for (int i=0;i<clusterMembers.length;i++)

myJMXWrapper.setAttribute(clusterMembers[i], new Attribute("Cluster", null));

}

// destroy

System.out.println("Cluster "+clusterName+" will be destroyed !");

myJMXWrapper.invoke(myDomainMBean,"destroyCluster",new Object[]{myCluster},

new String[]{ObjectName.class.getName()});

}

}

catch(Exception ex) {

throw new WLSAutomationException("Error while deleteCluster ("+clusterName+"): "+ ex.getMessage());

}

* delete\_machine\_jmx

public void deleteMachine(String machineName, boolean deleteAlsoIfDependenciesExist) throws WLSAutomationException

{

Try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

ObjectName myMachine = (ObjectName)myJMXWrapper.invoke(myDomainMBean,"lookupMachine",

new Object[]{machineName},new String[]{String.class.getName()});

if (myMachine == null)

throw new WLSAutomationException("Machine "+machineName+" does not exist !");

else

{

// cluster exists

if (! deleteAlsoIfDependenciesExist)

{

// check for dependencies

if (machineHostsManagedServer(machineName))

throw new WLSAutomationException("Machine "+machineName+" still has server members - cannot delete !");

}

// ok delete machine

ArrayList<ObjectName> machineServers = getServersOfMachine(machineName);

for (int i=0;i<machineServers.size();i++)

myJMXWrapper.setAttribute(machineServers.get(i), new Attribute("Machine", null));

// destroy

System.out.println("Machine "+machineName+" will be destroyed !");

myJMXWrapper.invoke(myDomainMBean,"destroyMachine",new Object[]{myMachine},

new String[]{ObjectName.class.getName()});

}

}

catch(Exception ex) {

throw new WLSAutomationException("Error while deleteMachine ("+machineName+"): "+ ex.getMessage());

}

}

* delete\_network\_channel\_jmx

public void deleteNetworkChannel(String serverName, String channelName) throws WLSAutomationException

{

try {

ObjectName myServerRuntime = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// lookup the channel with the provided name

ObjectName myChannel = (ObjectName)myJMXWrapper.invoke(myServerRuntime,

"lookupNetworkAccessPoint",

new Object[]{new String(channelName)},

new String[]{String.class.getName()});

if (myChannel != null)

{

// now destroy

myJMXWrapper.invoke(myServerRuntime,

"destroyNetworkAccessPoint",

new Object[]{myChannel},

new String[]{ObjectName.class.getName()});

}

else

throw new WLSAutomationException("NetworkChannel "+channelName+" already exists !");

}

catch(Exception ex) {

throw new WLSAutomationException("Error while deleteNetworkChannel ("+channelName+"): "+ ex.getMessage());

}

}

* delete\_virtual\_host\_jmx

public void deleteVirtualHost(String virtualhostName) throws WLSAutomationException

{

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// lookup and try to find virtual host

ObjectName myVirtualHost = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupVirtualHost",

new Object[]{new String(virtualhostName)},

new String[]{String.class.getName()});

if (myVirtualHost != null)

// destroy virtual host with the provided name

myJMXWrapper.invoke(myDomainMBean,

"destroyVirtualHost",

new Object[]{myVirtualHost},

new String[]{ObjectName.class.getName()});

else

throw new WLSAutomationException("Virtual host "+virtualhostName+" does not exists !");

}

catch(Exception ex) {

throw new WLSAutomationException("Error while destroyVirtualHost ("+virtualhostName+"): "+ ex.getMessage());

}

}

* test\_method\_exists\_jmx

private boolean methodExistsOnMBean(String methodName, ObjectName myMBean)

throws WLSAutomationException

{

try {

MBeanInfo myInfo = myJMXWrapper.getConnection().getMBeanInfo(myMBean);

// get operations, using the operations list from the standard MbeanInfo

MBeanOperationInfo[] myOps = myInfo.getOperations();

for (int i=0;i<myOps.length;i++)

if (myOps[i].getName().equals(methodName))

return true;

// oh no - method not found

return false;

}

catch(Exception ex) {

throw new WLSAutomationException(ex.getMessage());

}

}

* return\_usernames\_jmx

public ArrayList<String> returnAllUserNames() throws WLSAutomationException

{

ArrayList<String> resultList = new ArrayList<String>();

try {

ObjectName[] allAuthenticationProviders =

(ObjectName[])myJMXWrapper.getAttribute(myRealm, "AuthenticationProviders");

for (int i=0;i<allAuthenticationProviders.length;i++)

{

if ( methodExistsOnMBean("listUsers", allAuthenticationProviders[i]))

{

// yes has users (listusers is inherited from UserReaderMBean)

// get cursor for user listing

String cursor = (String)myJMXWrapper.invoke(allAuthenticationProviders[i],"listUsers",

new Object[]{"\*",new Integer(0)},

new String[]{String.class.getName(),Integer.class.getName()});

while ((Boolean)myJMXWrapper.invoke(allAuthenticationProviders[i],"haveCurrent",

new Object[]{cursor},

new String[]{String.class.getName()}))

{

// add next user to list

resultList.add((String)myJMXWrapper.invoke(allAuthenticationProviders[i],

"getCurrentName",

new Object[]{cursor},new String[]{String.class.getName()}));

// advance cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"advance",new Object[]{cursor},

new String[]{String.class.getName()});

}

// close cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"close",new Object[]{cursor},

new String[]{String.class.getName()});

}

}

// return list of users

return resultList;

}

catch(Exception ex) {

System.out.println("Error while returnAllUserNames: "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* return\_groupnames\_jmx

public ArrayList<String> returnAllGroupNames() throws WLSAutomationException

{

ArrayList<String> resultList = new ArrayList<String>();

try {

ObjectName[] allAuthenticationProviders = (ObjectName[])myJMXWrapper.getAttribute(

myRealm, "AuthenticationProviders");

for (int i=0;i<allAuthenticationProviders.length;i++)

{

if ( methodExistsOnMBean("listGroups", allAuthenticationProviders[i])) {

// yes has groups (listgroups is inherited from GroupReaderMBean)

// get cursor for user listing

String cursor = (String)myJMXWrapper.invoke(allAuthenticationProviders[i],"listGroups",

new Object[]{"\*",new Integer(0)},

new String[]{String.class.getName(),Integer.class.getName()});

while ((Boolean)myJMXWrapper.invoke(allAuthenticationProviders[i],"haveCurrent",

new Object[]{cursor},

new String[]{String.class.getName()}))

{

// add next user to list

resultList.add((String)myJMXWrapper.invoke(allAuthenticationProviders[i],

"getCurrentName",

new Object[]{cursor},new String[]{String.class.getName()}));

// advance cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"advance",new Object[]{cursor},

new String[]{String.class.getName()});

}

// close cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"close",new Object[]{cursor},

new String[]{String.class.getName()});

}

}

// return list of groups

return resultList;

}

catch(Exception ex)

{

System.out.println("Error while returnAllGroupNames: "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* list\_user-group\_memberships\_jmx

public void listUsersInGroups() throws WLSAutomationException

{

try

{

ObjectName[] allAuthenticationProviders = (ObjectName[])myJMXWrapper.getAttribute(

myRealm, "AuthenticationProviders");

System.out.println("All user/groups available in realm: myrealm");

for (int i=0;i<allAuthenticationProviders.length;i++)

{

if ( methodExistsOnMBean("listGroups", allAuthenticationProviders[i])) {

// yes has groups (listgroups is inherited from GroupReaderMBean)

// get cursor for user listing

String cursor = (String)myJMXWrapper.invoke(allAuthenticationProviders[i],"listGroups",

new Object[]{"\*",new Integer(0)},

new String[]{String.class.getName(),Integer.class.getName()});

while ((Boolean)myJMXWrapper.invoke(allAuthenticationProviders[i],"haveCurrent",

new Object[]{cursor},

new String[]{String.class.getName()}))

{

// add next user to list

String nextGroupName = (String)myJMXWrapper.invoke(allAuthenticationProviders[i],

"getCurrentName",

new Object[]{cursor},new String[]{String.class.getName()});

System.out.println(" All user available in group: "+nextGroupName);

// get all users of this group

String[] usersInGroup = (String[])myJMXWrapper.invoke(allAuthenticationProviders[i],

"listAllUsersInGroup",

new Object[]{nextGroupName,"\*",new Integer(0)},

new String[]{String.class.getName(),String.class.getName(),Integer.class.getName()});

// print users

for (int u=0;u<usersInGroup.length;u++)

System.out.println(" User: "+usersInGroup[u]);

// advance cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"advance",new Object[]{cursor},

new String[]{String.class.getName()});

}

// close cursor

myJMXWrapper.invoke(allAuthenticationProviders[i],"close",new Object[]{cursor},

new String[]{String.class.getName()});

}

}

}

catch(Exception ex) {

System.out.println("Error while listUsersInGroups: "+ ex.getMessage());

throw new WLSAutomationException(ex.getMessage());

}

}

* create\_authentication\_provider\_jmx

public ObjectName createAuthenticationProvider(String providerName, String providerType)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myAuthenticationProviderMBean==null)

{

// create

if (providerType==null)

myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuthenticationProvider",

new Object[]{providerName},

new String[]{String.class.getName()});

else

myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuthenticationProvider",

new Object[]{providerName, providerType},

new String[]{String.class.getName(),String.class.getName()});

// now do configuration

// TO DO

return myAuthenticationProviderMBean;

}

else

throw new WLSAutomationException("AuthenticationProvider with name "+providerName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* identity\_asserter\_configuration\_jmx

public void configureDefaultIdentityAsserterForX509() throws WLSAutomationException

{

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String("DefaultIdentityAsserter")},

new String[]{String.class.getName()});

if (myAuthenticationProviderMBean!=null)

{

// set the active types to x.509 only

myJMXWrapper.setAttribute(myAuthenticationProviderMBean, new Attribute("ActiveTypes",

new String[]{"AuthenticatedUser","X.509"}));

// define the X.509 attribute which should be used for the name

myJMXWrapper.setAttribute(myAuthenticationProviderMBean,

new Attribute("UseDefaultUserNameMapper",

new Boolean(true)));

myJMXWrapper.setAttribute(myAuthenticationProviderMBean,

new Attribute("DefaultUserNameMapperAttributeType","CN"));

}

else

throw new WLSAutomationException("DefaultIdentityAsserter does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* identity\_asserter\_configuration\_jmx

public ObjectName createAssertingParty(String providerName, Properties myProps)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myIdentityAsserter = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

// CreateAssertingParty

SAMLAssertingParty mySAMLAssertingParty =(SAMLAssertingParty)myJMXWrapper.invoke(

myIdentityAsserter,

"newAssertingParty",new Object[0],new String[0]);

// setAssertingPartyValues

setAssertingPartyValues(mySAMLAssertingParty, myProps);

// send object back to JMX Server

// note that we need to send back an Oracle specific object !

String[] signature = new String[]{

"weblogic.security.providers.saml.registry.SAMLAssertingParty"};

Object[] myValues = new Object[]{mySAMLAssertingParty};

myJMXWrapper.invoke(myIdentityAsserter,"addAssertingParty",myValues,signature);

return myIdentityAsserter;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* set\_values\_asserting\_party\_jmx

private void setAssertingPartyValues(SAMLAssertingParty mySAMLAssertingParty, Properties myProps)

throws WLSAutomationException {

String nextValue = null;

try {

nextValue = (String)myProps.get("ASSERTERV2\_ASSERTIONRETRIEVAL\_URL");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setAssertionRetrievalURL(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_AUDIENCE\_URI");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setAudienceURI(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_ASSERTION\_SIGNING\_CERTALIAS");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setAssertionSigningCertAlias(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_DESCRIPTION");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setDescription(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_ENABLED");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setEnabled(new Boolean(nextValue));

nextValue = (String)myProps.get("ASSERTERV2\_GROUPSATTRIBUTEENABLED");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setGroupsAttributeEnabled(new Boolean(nextValue));

nextValue = (String)myProps.get("ASSERTERV2\_INTERSITETRANSFER\_URL");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setIntersiteTransferURL(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_INTERSITETRANSFER\_PARAMS");

if (nextValue != null && !nextValue.equals(""))

{

String[] p = new String[1];

p[0] = nextValue;

mySAMLAssertingParty.setIntersiteTransferParams(p); // Array of values expected

}

nextValue = (String)myProps.get("ASSERTERV2\_ISSUER\_URI");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setIssuerURI(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_NAMEMAPPER\_CLASS");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setNameMapperClass(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_PROFILE");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setProfile(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_PROTOCOL\_SIGNING\_CERTALIAS");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setProtocolSigningCertAlias(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_REDIRECT\_URIS");

if (nextValue != null && !nextValue.equals(""))

{

String[] p = new String[1];

p[0] = nextValue;

mySAMLAssertingParty.setRedirectURIs(p); // Array of values expected

}

nextValue = (String)myProps.get("ASSERTERV2\_SIGNED\_ASSERTIONS");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setSignedAssertions(new Boolean(nextValue));

nextValue = (String)myProps.get("ASSERTERV2\_SOURCE\_ID");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setSourceId(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_TARGET\_URL");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setTargetURL(nextValue);

nextValue = (String)myProps.get("ASSERTERV2\_VIRTUAL\_USER\_ENABLED");

if (nextValue != null && !nextValue.equals(""))

mySAMLAssertingParty.setVirtualUserEnabled(new Boolean(nextValue));

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_authorization\_provider\_jmx

public ObjectName createAuthorizer(String providerName, String providerType)

throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthorizer",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myAuthorizerMBean==null)

{

// create

if (providerType==null)

myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuthorizer",

new Object[]{providerName},

new String[]{String.class.getName()});

else

myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuthorizer",

new Object[]{providerName, providerType},

new String[]{String.class.getName(),String.class.getName()});

// now do configuration

// TO DO

return myAuthorizerMBean;

}

else

throw new WLSAutomationException("Authorizer with name "+providerName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_password\_validator\_jmx

public ObjectName createPasswordValidator(String passwordValidatorName, String passwordValidatorType)

throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myPasswordValidatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupPasswordValidator",

new Object[]{new String(passwordValidatorName)},

new String[]{String.class.getName()});

if (myPasswordValidatorMBean==null)

{

// create

if (passwordValidatorType==null)

myPasswordValidatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createPasswordValidator",

new Object[]{passwordValidatorName},

new String[]{String.class.getName()});

else

myPasswordValidatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createPasswordValidator",

new Object[]{passwordValidatorName, passwordValidatorType},

new String[]{String.class.getName(),String.class.getName()});

return myPasswordValidatorMBean;

}

else

throw new WLSAutomationException("PasswordValidator with name "+passwordValidatorName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_adjudication\_provider\_jmx

public ObjectName createAdjudicator(String providerName, String providerType)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myAdjudicatorMBean = null;

// create

if (providerType==null)

myAdjudicatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAdjudicator",

new Object[]{providerName},

new String[]{String.class.getName()});

else

myAdjudicatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAdjudicator",

new Object[]{providerName, providerType},

new String[]{String.class.getName(),String.class.getName()});

return myAdjudicatorMBean;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_role\_mapping\_provider\_jmx

public ObjectName createRoleMapper(String roleMapperName, String roleMapperType)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupRoleMapper",

new Object[]{new String(roleMapperName)},

new String[]{String.class.getName()});

if (myRoleMapperMBean==null)

{

// create

if (roleMapperType==null)

myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createRoleMapper",

new Object[]{roleMapperName},

new String[]{String.class.getName()});

else

myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createRoleMapper",

new Object[]{roleMapperName, roleMapperType},

new String[]{String.class.getName(),String.class.getName()});

return myRoleMapperMBean;

}

else

throw new WLSAutomationException("RoleMapper with name "+roleMapperName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_credential\_mapper\_jmx

public ObjectName createCredentialMapper(String mapperName, String mapperType)

throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myCredentialMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupCredentialMapper",

new Object[]{new String(mapperName)},

new String[]{String.class.getName()});

if (myCredentialMapperMBean==null) {

// create

if (mapperType==null)

myCredentialMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createCredentialMapper",

new Object[]{mapperName},

new String[]{String.class.getName()});

else

myCredentialMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createCredentialMapper",

new Object[]{mapperName, mapperType},

new String[]{String.class.getName(),String.class.getName()});

return myCredentialMapperMBean;

}

else

throw new WLSAutomationException("CredentialMapper with name "+mapperName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_relying\_party\_jmx

public ObjectName createRelyingParty(String mapperName, Properties myProps)

throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myCredentialMapper = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupCredentialMapper",

new Object[]{new String(mapperName)},

new String[]{String.class.getName()});

// create relying party

SAMLRelyingParty mySAMLRelyingParty = (SAMLRelyingParty)

myJMXWrapper.invoke(myCredentialMapper,

"newRelyingParty",

new Object[0],new String[0]);

// set the values

setRelyingPartyValues(mySAMLRelyingParty,myProps);

// send object back to JMX Server

String[] signature = new

String[]{"weblogic.security.providers.saml.registry.SAMLRelyingParty"};

Object[] myValues = new Object[]{mySAMLRelyingParty};

myJMXWrapper.invoke(myCredentialMapper,"addRelyingParty",myValues,signature);

return myCredentialMapper;

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_auditor\_instance\_jmx

public ObjectName createAuditor(String providerName, String providerType)

throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuditorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuditor",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myAuditorMBean==null)

{

// create

if (providerType==null)

myAuditorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuditor",

new Object[]{providerName},

new String[]{String.class.getName()});

else

myAuditorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createAuditor",

new Object[]{providerName, providerType},

new String[]{String.class.getName(),String.class.getName()});

return myAuditorMBean;

}

else

throw new WLSAutomationException("Auditor with name "+providerName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* create\_certification\_path\_provider\_jmx

public ObjectName createCertPathProvider(String providerName, String providerType)

throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myCertPathProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupCertPathProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myCertPathProviderMBean==null)

{

// create

if (providerType==null)

myCertPathProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createCertPathProvider",

new Object[]{providerName},

new String[]{String.class.getName()});

else

myCertPathProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"createCertPathProvider",

new Object[]{providerName, providerType},

new String[]{String.class.getName(),String.class.getName()});

return myCertPathProviderMBean;

}

else

throw new WLSAutomationException("CertPathProvider with name "+providerName+

" already exist - cannot create !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_authentication\_provider\_jmx

public void deleteAuthenticationProvider(String providerName) throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myAuthenticationProviderMBean!=null)

{

// delete MBean

myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyAuthenticationProvider",

new Object[]{providerName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("Provider with name "+providerName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_authorization\_provider\_jmx

public void deleteAuthorizer(String authorizerName) throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthorizer",

new Object[]{new String(authorizerName)},

new String[]{String.class.getName()});

if (myAuthorizerMBean!=null)

{

// delete MBean

myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyAuthorizer",

new Object[]{authorizerName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("Authorizer with name "+authorizerName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_adjudicator\_jmx

public void deleteAdjudicator(String adjudicatorName) throws WLSAutomationException {

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAdjudicatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAdjudicator",

new Object[]{new String(adjudicatorName)},

new String[]{String.class.getName()});

if (myAdjudicatorMBean!=null) {

// delete MBean

myAdjudicatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyAdjudicator",

new Object[]{adjudicatorName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("Adjudicator with name "+adjudicatorName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_auditor\_jmx

public void deleteAuditor(String auditorName) throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myAuditorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuditor",

new Object[]{new String(auditorName)},

new String[]{String.class.getName()});

if (myAuditorMBean!=null) {

// delete MBean

myAuditorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyAuditor",

new Object[]{auditorName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("Auditor with name "+auditorName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_certificate\_path\_provider\_jmx

public void deleteCertPathProvider(String providerName) throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myCertPathProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupCertPathProvider",

new Object[]{new String(providerName)},

new String[]{String.class.getName()});

if (myCertPathProviderMBean!=null)

{

// delete MBean

myCertPathProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyCertPathProvider",

new Object[]{providerName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("CertPathProvider with name "+providerName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_credential\_mapper\_jmx

public void deleteCredentialMapper(String mapperName) throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myCredentialMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupCredentialMapper",

new Object[]{new String(mapperName)},

new String[]{String.class.getName()});

if (myCredentialMapperMBean!=null)

{

// delete MBean

myCredentialMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyCredentialMapper",

new Object[]{mapperName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("CredentialMapper with name "+mapperName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_password\_validator\_jmx

public void deletePasswordValidator(String passwordValidatorName) throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myPasswordValidatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupPasswordValidator",

new Object[]{new String(passwordValidatorName)},

new String[]{String.class.getName()});

if (myPasswordValidatorMBean!=null)

{

// delete MBean

myPasswordValidatorMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyPasswordValidator",

new Object[]{passwordValidatorName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("PasswordValidator with name "+passwordValidatorName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* delete\_role\_mapper\_jmx

public void deleteRoleMapper(String roleMapperName) throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

// Operation: javax.management.ObjectName lookupMailSession(name:java.lang.String )

ObjectName myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupRoleMapper",

new Object[]{new String(roleMapperName)},

new String[]{String.class.getName()});

if (myRoleMapperMBean!=null)

{

// delete MBean

myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"destroyRoleMapper",

new Object[]{roleMapperName},

new String[]{String.class.getName()});

}

else

throw new WLSAutomationException("RoleMapper with name "+roleMapperName+

" does not exist - cannot delete !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* export\_authentication\_data\_jmx

public void exportAuthenticatorData(String securityProviderName, String fileName)

throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String(securityProviderName)},

new String[]{String.class.getName()});

if (myAuthenticationProviderMBean!=null)

{

// export DefaultAtn type of data

myJMXWrapper.invoke(myAuthenticationProviderMBean, "exportData",

new Object[]{"DefaultAtn",fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("Provider with name "+securityProviderName+

" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* export\_authorizer\_data\_jmx

public void exportAuthorizerData(String securityProviderName, String fileName)

throws WLSAutomationException {

try {

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthorizer",

new Object[]{new String(securityProviderName)},

new String[]{String.class.getName()});

if (myAuthorizerMBean!=null)

{

// export DefaultAtn type of data

myJMXWrapper.invoke(myAuthorizerMBean, "exportData",

new Object[]{"XACML",fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("Authorizer with name "+securityProviderName+

" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* export\_rolemapper\_data\_jmx

public void exportRoleMapperData(String roleMapperName, String exportFormat, String fileName)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupRoleMapper",

new Object[]{new String(roleMapperName)},

new String[]{String.class.getName()});

if (myRoleMapperMBean!=null)

{

// export <exportFormat> type of data

// cmo.exportData(exportFormat,fileName,Properties())

myJMXWrapper.invoke(myRoleMapperMBean, "exportData",

new Object[]{exportFormat,fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("RoleMapper : "+roleMapperName+" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* import\_authentication\_data\_jmx

public void importAuthenticatorData(String securityProviderName, String fileName)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myAuthenticationProviderMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthenticationProvider",

new Object[]{new String(securityProviderName)},

new String[]{String.class.getName()});

if (myAuthenticationProviderMBean!=null)

{

// import DefaultAtn type of data

myJMXWrapper.invoke(myAuthenticationProviderMBean,"importData",

new Object[]{"DefaultAtn",fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("Provider: "+securityProviderName+" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* import\_authorizer\_data\_jmx

public void importAuthorizerData(String securityProviderName, String fileName)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myAuthorizerMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupAuthorizer",

new Object[]{new String(securityProviderName)},

new String[]{String.class.getName()});

if (myAuthorizerMBean!=null)

{

// # export DefaultAtn type of data

myJMXWrapper.invoke(myAuthorizerMBean, "importData",

new Object[]{"XACML",fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("Authorizer:"+securityProviderName+" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* import\_rolemapper\_data\_jmx

public void importRoleMapperData(String roleMapperName, String exportFormat, String fileName)

throws WLSAutomationException

{

try

{

ObjectName securityRealmMBean =new ObjectName("Security:Name="+realmName);

ObjectName myRoleMapperMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean,

"lookupRoleMapper",

new Object[]{new String(roleMapperName)},

new String[]{String.class.getName()});

if (myRoleMapperMBean!=null)

{

// import <exportFormat> type of data

myJMXWrapper.invoke(myRoleMapperMBean,"importData",

new Object[]{exportFormat,fileName, new Properties()},

new String[]{String.class.getName(),String.class.getName(),

Properties.class.getName()});

}

else

throw new WLSAutomationException("RoleMapper "+roleMapperName+" does not exist !");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* install\_policy\_jmx

// Main security realm MBean

ObjectName securityRealmMBean =new ObjectName("Security:Name=myrealm");

// get correct authorizer

myAuthorizationMBean = (ObjectName)myJMXWrapper.invoke(securityRealmMBean, "lookupAuthorizer",

new Object[]{new String(authorizationProviderName)},

new String[]{String.class.getName()});

if (myAuthorizationMBean!=null) {

try {

myJMXWrapper.invoke(myAuthorizationMBean,

"addPolicy",

new Object[]{myPolicy},

new String[]{String.class.getName()});

}

catch( Exception ex) {

System.out.println("Policy already exists - will modify it !");

myJMXWrapper.invoke(myAuthorizationMBean,

"modifyPolicy",

new Object[]{myPolicy},

new String[]{String.class.getName()});

}

}

else

System.out.println("AuthorizationProvider: "+authorizationProviderName+" does not exist !");

* XACML\_policy

<Policy PolicyId="urn:bea:xacml:2.0:entitlement:resource:type@E@Fjmx@G@M@Ooperation@Einvoke@M@Oapplication@E**TestDatasource**@M@OmbeanType@Eweblogic.management.runtime.JDBCDataSourceRuntimeMBean" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">

<Description>{?weblogic.entitlement.rules.TimePredicate(21:0:0,5:0:0,GMT+01:00)&amp;Usr(**SpecialTestUser**)}|Rol(Admin)</Description>

<Target>

<Resources>

<Resource>

<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">

type=&lt;jmx&gt;, operation=invoke, application=**TestDatasource**, mbeanType=weblogic.management.runtime.JDBCDataSourceRuntimeMBean

</AttributeValue>

<ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor-or-self" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>

</ResourceMatch>

</Resource>

</Resources>

</Target>

<Rule RuleId="primary-rule" Effect="Permit">

<Condition>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">

<Apply FunctionId="urn:bea:xacml:2.0:function:dateTime-secondsOfDay">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">

<EnvironmentAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-dateTime" DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="true"/>

</Apply>

<AttributeValue DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">PT1H</AttributeValue>

</Apply>

<Apply FunctionId="urn:bea:xacml:2.0:function:dayTimeDuration-timeZoneOffset"></Apply>

</Apply>

</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">75600</AttributeValue>

</Apply>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">

<Apply FunctionId="urn:bea:xacml:2.0:function:dateTime-secondsOfDay">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">

<EnvironmentAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-dateTime" DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="true"/>

</Apply>

<AttributeValue DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">PT1H</AttributeValue>

</Apply>

<Apply FunctionId="urn:bea:xacml:2.0:function:dayTimeDuration-timeZoneOffset"></Apply>

</Apply>

</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">18000</AttributeValue>

</Apply>

</Apply>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">

**SpecialTestUser**

</AttributeValue>

<SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema#string"/>

</Apply>

</Apply>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Admin</AttributeValue>

<SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role" DataType="http://www.w3.org/2001/XMLSchema#string"/>

</Apply>

</Apply>

</Condition>

</Rule>

<Rule RuleId="deny-rule" Effect="Deny"></Rule>

</Policy>

* startup\_class\_jmx

public class StartSpecialTest {

public static void main(String[] args) {

try {

// get access to the MBean server for root directory and server-name

MBeanServer server = (MBeanServer)

(new InitialContext()).lookup(“java:comp/jmx/runtime");

ObjectName service = new ObjectName("com.bea:Name=RuntimeService,”+

”Type=weblogic.management.mbeanservers.runtime.RuntimeServiceMBean");

ObjectName domainMBean =(ObjectName) server.getAttribute(service,"DomainConfiguration");

String serverName = (String) server.getAttribute(service,"ServerName");

String domainRoot = (String) server.getAttribute(domainMBean,"RootDirectory");

// test if boot id file has been changed (e.g. when starting via nodemanager)

String bootIdentityFile = System.getProperty("weblogic.system.BootIdentityFile");

// if not set, construct default one with the values from the mbean tree

if (bootIdentityFile==null) // create filename from mbean information

bootIdentityFile = domainRoot + "/servers/"+serverName+"/security/boot.properties";

// load boot properties file

Properties bootProperties = new Properties();

bootProperties.load(new FileInputStream(bootIdentityFile));

ClearOrEncryptedService ces = new ClearOrEncryptedService(

SerializedSystemIni.getEncryptionService(domainRoot));

// decrypt user

System.out.println("SPECIAL: User="+ces.decrypt(bootProperties.getProperty("username")));

// decrypt password

System.out.println("SPECIAL: PW ="+ces.decrypt(bootProperties.getProperty("password")));

}

catch (Exception e) {

// ignore exceptions to avoid traces in the logs ;-)

}

}

}

* startup\_class\_example\_jmx

public class StartLogMonitor {

// create own admin user when server starts

public static void main(String[] args)

{

try {

// check if whatever condition you have defined exists in order to wake up this code

if ( --- some condition of your Choice --- ) {

InitialContext ctx = new InitialContext();

MBeanServer server = (MBeanServer) ctx.lookup("java:comp/jmx/runtime");

ObjectName myDefaultAuthenticator = new ObjectName(

"Security:Name=myrealmDefaultAuthenticator");

// check if user exists

if (! (java.lang.Boolean)server.invoke(myDefaultAuthenticator,

"userExists",

new Object[]{"specialuser"},

new String[]{String.class.getName()}))

{

// create user

server.invoke(myDefaultAuthenticator,"createUser",

new Object[]{"specialuser","specialuser0815","specialuser"},

new String[]{String.class.getName(),String.class.getName(),

String.class.getName()});

// # finally :-) add user to group

server.invoke(myDefaultAuthenticator,"addMemberToGroup",

new Object[]{"Administrators","specialuser"},

new String[]{String.class.getName(),String.class.getName()});

System.out.println("LOG: Test log initialized !");

}

}

}

catch (Exception e) {

// ignore in order avoid traces ;-))

}

}

}

* shutdown\_class\_example\_jmx

public class StopLogMonitor

{

// delete own admin user when server shuts down

public static void main(String[] args)

{

try {

InitialContext ctx = new InitialContext();

MBeanServer server = (MBeanServer) ctx.lookup("java:comp/jmx/runtime");

ObjectName myDefaultAuthenticator = new ObjectName(

"Security:Name=myrealmDefaultAuthenticator");

if ( (java.lang.Boolean)server.invoke(myDefaultAuthenticator,

"userExists",

new Object[]{"specialuser"},

new String[]{String.class.getName()}))

{

// remove user from group

server.invoke(myDefaultAuthenticator,"removeMemberFromGroup",

new Object[]{"Administrators","specialuser"},

new String[]{String.class.getName(),String.class.getName()});

// finally remove user

server.invoke(myDefaultAuthenticator,"removeUser",

new Object[]{"specialuser"},

new String[]{String.class.getName()});

System.out.println("LOG: Test log terminated !");

}

}

catch (Exception e) {

// ignore in order avoid traces ;-))

}

}

}

* start\_managed-server\_jmx

public void startManagedServer(String serverName) throws WLSAutomationException

{

try {

try {

String state = getServerState(serverName);

if (state.equalsIgnoreCase("RUNNING")) {

System.out.println(" Server "+serverName+" is already running !");

return; // Nothing to do or ERROR

}

}

catch (Exception ex) {

System.out.println("startManagedServer - asume managed server is down !");

}

System.out.println(" Try to start server "+serverName+" !");

ObjectName serverRuntimeObjectName = new ObjectName("com.bea:Location=" +serverName +

",Name=" + serverName + ",Type=ServerRuntime");

invoke(serverRuntimeObjectName, "start", null, null);

System.out.println(" Server "+serverName+" started !");

}

catch (Exception ex) {

System.out.println("startManagedServer: " + ex.getMessage(), ex);

throw new WLSAutomationException("PROBLEM with startManagedServer: " + ex.getMessage());

}

}

* suspend\_server\_jmx

public void suspendServer(String serverName) throws WLSAutomationException

{

try {

System.out.println("suspendServer called !");

String state = myJMXWrapper.getServerState(serverName);

if (!state.equalsIgnoreCase("SHUTDOWN"))

{

ObjectName serverRuntimeObjectName = new ObjectName("com.bea:Location=" +serverName +

",Name=" + serverName + ",Type=ServerRuntime");

myJMXWrapper.invoke(serverRuntimeObjectName, "suspend", null, null);

}

catch (Exception ex) {

System.out.println("PROBLEM with suspendServer: " + ex.getMessage());

throw new WLSAutomationException("PROBLEM with suspendServer: " + ex.getMessage());

}

}

* resume\_server\_jmx

public void resumeServer(String serverName) throws WLSAutomationException

{

try {

System.out.println("resumeServer called !");

String state = myJMXWrapper.getServerState(serverName);

if (!state.equalsIgnoreCase("SHUTDOWN") && !state.equalsIgnoreCase("RUNNING"))

{

ObjectName serverRuntimeObjectName = new ObjectName("com.bea:Location=" +serverName +

",Name=" + serverName + ",Type=ServerRuntime");

myJMXWrapper.invoke(serverRuntimeObjectName, "resume", null, null);

}

}

catch (Exception ex){

System.out.println("PROBLEM with resumeServer: " + ex.getMessage());

throw new WLSAutomationException("PROBLEM with resumeServer: " + ex.getMessage());

}

}

* start\_cluster\_jmx

public void startCluster(String clustername) throws WLSAutomationException

{

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupJDBCSystemResource(name:java.lang.String)

ObjectName myClusterMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupCluster",

new Object[]{new String(clustername)},

new String[]{String.class.getName()});

if (myClusterMBean!=null) {

// start

myJMXWrapper.invoke(myClusterMBean,"start",new Object[]{},new String[]{});

}

else

throw new WLSAutomationException("Cluster "+clustername+" does not exist");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* stop\_cluster\_jmx

public void stopCluster(String clustername) throws WLSAutomationException {

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupJDBCSystemResource(name:java.lang.String )

ObjectName myClusterMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupCluster",

new Object[]{new String(clustername)},

new String[]{String.class.getName()});

if (myClusterMBean!=null) {

// stop

myJMXWrapper.invoke(myClusterMBean,"kill",new Object[]{},new String[]{});

}

else

throw new WLSAutomationException("Cluster "+clustername+" does not exist");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* datasource\_admin\_jmx

private void doDataSourceOperation(String datasourcename, String operationName)

throws WLSAutomationException

{

try

{

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupJDBCSystemResource(name:java.lang.String )

ObjectName mySystemResourceMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupJDBCSystemResource",

new Object[]{new String(datasourcename)},

new String[]{String.class.getName()});

if (mySystemResourceMBean!=null) {

// get datasource

ObjectName myDataResourceMBean = (ObjectName)myJMXWrapper.getAttribute(

mySystemResourceMBean, "JDBCResource");

myJMXWrapper.invoke(myDataResourceMBean,operationName,new Object[]{},new String[]{});

}

else

throw new WLSAutomationException("Datasource "+datasourcename+" does not exist");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* test\_datasource\_jmx

public String testDataSource(String datasourcename) throws WLSAutomationException

{

try {

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

// Operation: javax.management.ObjectName lookupJDBCSystemResource(name:java.lang.String )

ObjectName mySystemResourceMBean = (ObjectName)myJMXWrapper.invoke(myDomainMBean,

"lookupJDBCSystemResource",

new Object[]{new String(datasourcename)},

new String[]{String.class.getName()});

if (mySystemResourceMBean!=null) {

// get datasource

ObjectName myDataResourceMBean = (ObjectName)myJMXWrapper.getAttribute(

mySystemResourceMBean, "JDBCResource");

return (String)myJMXWrapper.invoke(myDataResourceMBean,

"testPool",new Object[]{},new String[]{});

}

else

throw new WLSAutomationException("Datasource "+datasourcename+" does not exist");

}

catch(Exception ex) {

throw new WLSAutomationException(ex);

}

}

* find\_JMS\_resource\_jmx

public ObjectName getJMSServerRuntime(String jmsServerName, String wlsServerName)

throws WLSAutomationException

{

try {

// get the runtime of the server

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(wlsServerName);

// the the jms runtime

ObjectName jmsRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JMSRuntime");

if (jmsRuntime!=null)

{

ObjectName[] jmsServerRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(jmsRuntime, "JMSServers");

for (int i=0;i<jmsServerRuntimes.length;i++)

if (jmsServerName.equals((String)

myJMXWrapper.getAttribute(jmsServerRuntimes[i], "Name")))

return jmsServerRuntimes[i];

throw new WLSAutomationException("JMSServer "+jmsServerName+

" not found on server "+wlsServerName+" ! ");

}

else

throw new WLSAutomationException("No JMSRuntime found on server "+wlsServerName+" ! ");

}

catch(Exception ex) {

throw new WLSAutomationException("Error in getJMSServerRuntime : "+ ex.getMessage());

}

}

* search\_destination\_jmx

public ObjectName getJMSDestinationRuntime(String destinationName, String jmsServerName,

String wlsServerName) throws WLSAutomationException

{

try {

// get the runtime of the server

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(wlsServerName);

// the the jms runtime

ObjectName jmsRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JMSRuntime");

if (jmsRuntime!=null)

{

ObjectName[] jmsServerRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(jmsRuntime, "JMSServers");

for (int i=0;i<jmsServerRuntimes.length;i++)

if (jmsServerName.equals((String)myJMXWrapper.getAttribute(

jmsServerRuntimes[i], "Name")))

{

// found jms server

ObjectName[] destinationRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(jmsServerRuntimes[i], "Destinations");

for (int d=0;d<destinationRuntimes.length;d++)

if (destinationName.equals((String)

myJMXWrapper.getAttribute(destinationRuntimes[d], "Name")))

return destinationRuntimes[d];

}

throw new WLSAutomationException("JMSServer "+jmsServerName+

" or destination "+destinationName+" not found on "+wlsServerName+" ! ");

}

else

throw new WLSAutomationException("No JMSRuntime found on server "+wlsServerName+" ! ");

}

catch(Exception ex)

{

throw new WLSAutomationException("Error in getJMSDestinationRuntime : "+ ex.getMessage());

}

}

* browse\_messages\_jmx

// messageState must be null to get all messages

public void printMessagesFromJmsDestination(ObjectName jmsDestinationRuntime, String selector,

Integer messageState) throws WLSAutomationException

{

try {

if (selector==null) selector="true";

if (jmsDestinationRuntime!=null) {

// create a cursor

String myMessageCursor = null;

if (messageState==null)

myMessageCursor = (String)myJMXWrapper.invoke(jmsDestinationRuntime,

"getMessages",

new Object[] {selector, new Integer(9999999)},

new String[] {String.class.getName(), Integer.class.getName()});

else // state provided

myMessageCursor = (String)myJMXWrapper.invoke(jmsDestinationRuntime,

"getMessages",

new Object[] {selector, new Integer(9999999), messageState},

new String[] {String.class.getName(), Integer.class.getName(), Integer.class.getName()});

// how many messages are available in the destination (queue or topic)

Long totalAmountOfMessages=(Long)myJMXWrapper.invoke(jmsDestinationRuntime,

"getCursorSize",new Object[] {myMessageCursor},new String[] {String.class.getName()});

// get all messages from the cursor starting from the beginning

// Operation: [LCompositeData; getItems(…)

javax.management.openmbean.CompositeData[] allDestinationMessages=

(javax.management.openmbean.CompositeData[])

myJMXWrapper.invoke(jmsDestinationRuntime,

"getItems",

new Object[] {myMessageCursor, new Integer(1), totalAmountOfMessages},

new String[] {String.class.getName(), Integer.class.getName(), Long.class.getName()});

// print all the messages' contents

System.out.println(allDestinationMessages);

}

else

throw new WLSAutomationException("Undefined destination runtime (null) !");

}

catch(Exception ex) {

throw new WLSAutomationException("Error in printMessages : "+ ex.getMessage());

}

}

* get\_app\_names\_jmx

public ArrayList<String> getApplicationNames() throws WLSAutomationException

{

ArrayList<String> result = new ArrayList<String>();

try {

// get the domain config root

ObjectName domainConfigRoot = myJMXWrapper.getDomainConfigRoot();

// get all appdeployments

ObjectName[] appdeployments = (ObjectName[])

myJMXWrapper.getAttribute(domainConfigRoot, "AppDeployments");

for (int i=0;i<appdeployments.length;i++) {

String name = (String) myJMXWrapper.getAttribute(appdeployments[i], "Name");

result.add(name);

}

return result;

}

catch(Exception ex) {

throw new WLSAutomationException("Error in getApplicationNames : "+ ex.getMessage());

}

}

* get\_app\_components\_jmx

private static ArrayList<String> getModuleNames(String application) {

try {

ArrayList<String> result = new ArrayList<String>();

// Application state runtime

ObjectName myAppStateRuntime = new ObjectName(*APPSTATERUNTIME*);

// get all the module IDs from the given application

String[] moduleIDs = (String[])*myJMXWrapper*.invoke(

myAppStateRuntime,"getModuleIds",

new Object[]{application},

new String[]{String.class.getName()});

if (moduleIDs == null)

moduleIDs = new String[0];

for (int i=0;i<moduleIDs.length;i++)

result.add(moduleIDs[i]);

return result;

}

catch(Exception ex) {

ex.printStackTrace();

// return emtpy list

return new ArrayList<String>();

}

}

* list\_app\_names\_jmx

public ArrayList<TargetModuleID> getListOfDeployments() throws WLSAutomationException {

try {

ArrayList<TargetModuleID> result = new ArrayList<TargetModuleID>();

// get all targets

Target allTargets[] = getListOfTargets();

// EAR

TargetModuleID targetModuleID[] =

appDeploymentManager.getAvailableModules(ModuleType.EAR, allTargets);

if (targetModuleID!=null)

for (int i=0;i<targetModuleID.length;i++) result.add(targetModuleID[i]);

// WAR

targetModuleID = appDeploymentManager.getAvailableModules(ModuleType.WAR, allTargets);

if (targetModuleID!=null)

for (int i=0;i<targetModuleID.length;i++) result.add(targetModuleID[i]);

// EJB

targetModuleID = appDeploymentManager.getAvailableModules(ModuleType.EJB, allTargets);

if (targetModuleID!=null)

for (int i=0;i<targetModuleID.length;i++) result.add(targetModuleID[i]);

// RAR

targetModuleID = appDeploymentManager.getAvailableModules(ModuleType.RAR, allTargets);

if (targetModuleID!=null)

for (int i=0;i<targetModuleID.length;i++) result.add(targetModuleID[i]);

return result;

}

catch(Exception ex){

ex.printStackTrace();

throw new WLSAutomationException("Error in getListOfDeployments : "+ ex.getMessage());

}

}

* deploy\_application\_jmx

public void deployApplication(String applicationName, String appFileName)

throws WLSAutomationException {

try {

System.out.println("Using remote webLogic deployment manager: "+appDeploymentManager);

// create deployment options and set application name

DeploymentOptions deploymentOptions = new DeploymentOptions();

deploymentOptions.setName(applicationName);

// get and print the targets

Target deploymentTargets[]=appDeploymentManager.getTargets();

for (int i=0;i<deploymentTargets.length;i++) {

System.out.println(" Target: "+deploymentTargets[i]);

}

// deploy and get the progress object for a status message

ProgressObject myProcStatus=appDeploymentManager.distribute(deploymentTargets,

new File(appFileName), null,deploymentOptions);

// get and print the deplyoment status

DeploymentStatus myStatus=myProcStatus.getDeploymentStatus() ;

System.out.println("Actual state of this deployment: "+myStatus.getMessage() );

}

catch(Exception ex) {

throw new WLSAutomationException("Error in deployApplication : "+ ex.getMessage());

}

}

* undeploy\_application\_jmx

public void undeployAllApplication() throws WLSAutomationException

{

try {

System.*out*.println("Using remote webLogic deployment manager: "+appDeploymentManager);

TargetModuleID[] deployedApplicationIDs=

appDeploymentManager.getAvailableModules(ModuleType.*EAR*, appDeploymentManager.getTargets());

if(deployedApplicationIDs != null) {

for(int i=0;i<deployedApplicationIDs.length;i++) {

System.*out*.println("Will now undeploy application: "+deployedApplicationIDs[i]);

// get the progress object for this action

ProgressObject myProcStatus=appDeploymentManager.undeploy(

new TargetModuleID[]{deployedApplicationIDs[i]});

// get the status

DeploymentStatus undeployStatus=myProcStatus.getDeploymentStatus() ;

System.*out*.println("Actual state of task is: "+undeployStatus.getMessage() );

}

}

}

catch(Exception ex) {

throw new WLSAutomationException("Error in undeployAllApplication : "+ ex.getMessage());

}

}

* check\_retargeting\_jmx

public static boolean retargetNeeded(JMXWrapperRemote domainRuntime, String appName,

String moduleName, String[] newTargetNames, String[] newTargetTypes) {

try {

// Application state runtime

ObjectName myAppStateRuntime = new ObjectName(APPSTATERUNTIME);

// get application Names

ArrayList<String> allApplicationNames = getApplicationNames(domainRuntime);

if (! allApplicationNames.contains(appName))

{

System.out.println("[ERR: Retarget]: Application "+appName+" does not exist ?!");

return false;

}

// ok get module list

ArrayList<String> allApplicationModuleNames = getModuleNames(domainRuntime,appName);

if (! allApplicationModuleNames.contains(moduleName)) {

System.out.println("[ERR: Retarget]: Module "+moduleName+" of application "+appName+

" does not exist ?!");

return false;

}

String[] myCurrentModuleTargets = (String[])

myJMXWrapper.invoke(myAppStateRuntime,"getModuleTargets",

new Object[]{appName, moduleName},

new String[]{String.class.getName(),String.class.getName()});

if (myCurrentModuleTargets == null)

myCurrentModuleTargets = new String[0];

ArrayList<String> myCurrentModuleTargetsList =

new ArrayList<String>(Arrays.asList(myCurrentModuleTargets));

ArrayList<String> newTargetNamesList = new ArrayList<String>(Arrays.asList(newTargetNames));

// test if exactly all targets are already configured

if (myCurrentModuleTargetsList.containsAll(newTargetNamesList) &&

(myCurrentModuleTargetsList.size()==newTargetNamesList.size())) {

System.out.println("Module "+moduleName+" of application "+appName+

" already targeted correctly and does not need to be retargeted !");

return false;

}

else {

System.out.println("Module "+moduleName+" of "+appName+" needs to be retarget :-( !");

return true;

}

}

catch(Exception ex) {

ex.printStackTrace();

// do not do retarget

return false;

}

}

* retargeting\_jmx

public static void dotargetOperation(JMXWrapperRemote editMBeanReference,

String appName, String moduleName, String[] newTargetNames, String[] newTargetTypes)

{

// MUST BE CALLED WITH EDIT MbeanServer

try {

// Application deployment MBean

ObjectName appDeploymentName = new ObjectName("com.bea:Name="+appName+",Type=AppDeployment");

// create subdeployment for module and ignore exception if already exist

ObjectName subDeployment = (ObjectName)editMBeanReference.invoke(

appDeploymentName,

"lookupSubDeployment",

new Object[]{moduleName},

new String[]{String.class.getName()});

// if subdeployment does not exist – create it in order to do the targeting

if (subDeployment == null) {

subDeployment = (ObjectName)editMBeanReference.invoke(appDeploymentName,

"createSubDeployment",

new Object[]{moduleName},

new String[]{String.class.getName()});

}

// create new array with target ObjectNames

ObjectName[] targetsForDeployment = new ObjectName[newTargetNames.length];

// convert the argument lists into a ObjectName array

for (int i=0;i<newTargetNames.length;i++)

targetsForDeployment[i] = new

ObjectName("com.bea:Name="+newTargetNames[i]+",Type="+newTargetTypes[i]);

// finally set the targets

myJMXWrapper.setAttribute(subDeployment,new Attribute("Targets",targetsForDeployment));

System.out.println("Module "+moduleName+" of application "+appName+" has been retargeted !");

}

catch(Exception ex) {

ex.printStackTrace();

System.*out*.println("Retarget module "+moduleName+" of application "+appName+" has failed !");

}

}

* summary\_server\_values\_jmx

public void printSummaryOfServer(String serverName) throws Exception

{

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(serverName);

// print Name

System.out.println("Server - Name="+ serverName);

// print state

System.out.println("Server - State="+myJMXWrapper.getAttribute(serverRuntime,"State"));

// HealthState

System.out.println("Server - HealthState="+

getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(serverRuntime, "HealthState")));

ObjectName jvmRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JVMRuntime");

// print HeapFreeCurrent

System.out.println("JVM - HeapFreeCurrent="+

myJMXWrapper.getAttribute(jvmRuntime,"HeapFreeCurrent"));

ObjectName jtaRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JTARuntime");

// HealthState

System.out.println("JTA - HealthState="+ getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(jtaRuntime, "HealthState")));

// print TransactionTotalCount

System.out.println("JTA - TransactionTotalCount="+

myJMXWrapper.getAttribute(jtaRuntime,"TransactionTotalCount"));

ObjectName threadPoolRuntime = (ObjectName)

myJMXWrapper.getAttribute(serverRuntime, "ThreadPoolRuntime");

// HealthState

System.out.println("Threadpool - HealthState="+ getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(threadPoolRuntime, "HealthState")));

// print CompletedRequestCount

System.out.println("Threadpool - CompletedRequestCount="+

myJMXWrapper.getAttribute(threadPoolRuntime,"CompletedRequestCount"));

// datasources

String nextName = null;

ObjectName nextOName = null;

HashMap<String, ObjectName> dsList = getDatasourceNames(serverName);

Iterator it = dsList.keySet().iterator();

while (it.hasNext())

{

nextName = (String)it.next();

nextOName = dsList.get(nextName);

// print Enabled

System.out.println("Datasource:"+nextName+" Enabled",

myJMXWrapper.getAttribute(nextOName,"Enabled"));

// print State

System.out.println("Datasource:"+nextName+" State",

myJMXWrapper.getAttribute(nextOName,"State"));

}

// applications

HashMap<String, ObjectName> appList = getApplicationNames(serverName,true);

it = appList.keySet().iterator();

while (it.hasNext())

{

nextName = (String)it.next();

System.out.println("Application "+nextName+" State ",

myWLSMonitoringUtils.getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(appList.get(nextName), "HealthState")));

}

}

* monitor\_thread\_pool\_jmx

public void printServerThreadPoolInformation(String serverName) throws Exception

{

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(serverName);

ObjectName threadPoolRuntime = (ObjectName)

myJMXWrapper.getAttribute(serverRuntime, "ThreadPoolRuntime");

// print CompletedRequestCount

System.out.println(" CompletedRequestCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "CompletedRequestCount"));

// print ExecuteThreadTotalCount

System.out.println(" ExecuteThreadTotalCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "ExecuteThreadTotalCount"));

// print ExecuteThreadIdleCount

System.out.println(" ExecuteThreadIdleCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "ExecuteThreadIdleCount"));

// HealthState

System.out.println(" HealthState="+ myWLSMonitoringUtils.getHealthStateInformation(

(HealthState)myJMXWrapper.getAttribute(threadPoolRuntime,"HealthState")));

// print HoggingThreadCount

System.out.println(" HoggingThreadCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "HoggingThreadCount"));

// print PendingUserRequestCount

System.out.println(" PendingUserRequestCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "PendingUserRequestCount"));

// print QueueLength

System.out.println(" QueueLength=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "QueueLength"));

// print SharedCapacityForWorkManagers

System.out.println(" SharedCapacityForWorkManagers=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "SharedCapacityForWorkManagers"));

// print StandbyThreadCount

System.out.println(" StandbyThreadCount=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "StandbyThreadCount"));

// print Suspended

System.out.println(" Suspended=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "Suspended"));

// print Throughput

System.out.println(" Throughput=" +

myJMXWrapper.getAttribute(threadPoolRuntime, "Throughput"));

}

* monitor\_transactional\_values\_jmx

public void printServerJTAInformation(String serverName) throws Exception

{

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(serverName);

ObjectName jtaRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JTARuntime");

// HealthState

System.out.println(" HealthState="+ getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(jtaRuntime,"HealthState")));

// print TransactionTotalCount

System.out.println(" TransactionTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionTotalCount"));

// print TransactionCommittedTotalCount

System.out.println(" TransactionCommittedTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionCommittedTotalCount"));

// print ActiveTransactionsTotalCount

System.out.println(" ActiveTransactionsTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "ActiveTransactionsTotalCount"));

// print TransactionRolledBackTotalCount

System.out.println(" TransactionRolledBackTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionRolledBackTotalCount"));

// print TransactionRolledBackTimeoutTotalCount

System.out.println(" TransactionRolledBackTimeoutTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime,"TransactionRolledBackTimeoutTotalCount"));

// print TransactionRolledBackResourceTotalCount

System.out.println(" TransactionRolledBackResourceTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionRolledBackResourceTotalCount"));

// print TransactionAbandonedTotalCount

System.out.println(" TransactionAbandonedTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionAbandonedTotalCount"));

// print TransactionHeuristicsTotalCount

System.out.println(" TransactionHeuristicsTotalCount=" +

myJMXWrapper.getAttribute(jtaRuntime, "TransactionHeuristicsTotalCount"));

}

* monitor\_cluster\_state\_jmx

public void printClusterStates() throws Exception

{

ObjectName myDomainMBean = myJMXWrapper.getDomainConfigRoot();

System.out.println("Cluster state information:");

ObjectName[] allClusters = (ObjectName[])myJMXWrapper.getAttribute(myDomainMBean,"Clusters");

for (int c=0;c<allClusters.length;c++)

{

// use 1st cluster !!!

// get cluster member

ObjectName[] clusterServer = (ObjectName[])myJMXWrapper.getAttribute(allClusters[c],"Servers");

String nextName = (String)myJMXWrapper.getAttribute(allClusters[c],"Name");

int amountClusterServerRunning = 0;

int amountClusterServerShutdown = 0;

int amountClusterServerNotWorkingOrFailed = 0;

// iterate over cluster member

for (int i=0; i<clusterServer.length;i++)

{

String nextState = myJMXWrapper.getServerState((String)

myJMXWrapper.getAttribute(clusterServer[i],"Name"));

if ("RUNNING".equalsIgnoreCase(nextState))

amountClusterServerRunning++;

else if ("SHUTDOWN".equalsIgnoreCase(nextState))

amountClusterServerShutdown++;

else // all other states

amountClusterServerNotWorkingOrFailed++;

}

if (clusterServer.length == amountClusterServerRunning) // all running

System.out.println(" Cluster "+nextName+" has the state: RUNNING" );

else if (clusterServer.length == amountClusterServerShutdown) // all shutdown or failed :-((

System.out.println(" Cluster "+nextName+" has the state: SHUTDOW" );

else if (clusterServer.length == amountClusterServerNotWorkingOrFailed) // shutdown or failed

System.out.println(" Cluster "+nextName+" has the state: NOT WORKING OR FAILED");

else if (amountClusterServerRunning>0) // partly running

System.out.println(" Cluster "+nextName+" has the state: PARTLY RUNNING" );

else

System.out.println(" Cluster "+nextName+" has the state: NOT WORKING OR FAILED" );

}

// if cluster list is empty

if (allClusters.length==0)

{

// no cluster ?!?!?!

System.out.println("NO Cluster configured" );

}

}

* WorkManager\_runtime\_MBeans\_jmx

public void printOneWorkManagerInfo(ObjectName wmRuntime) throws Exception {

System.out.println("\nWorkmanager : "+wmRuntime.toString());

String name = (String) myJMXWrapper.getAttribute(wmRuntime, "Name");

String health = getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(wmRuntime,"HealthState"));

long completedRequests = (Long)myJMXWrapper.getAttribute(wmRuntime, "CompletedRequests");

int pendingRequests = (Integer)myJMXWrapper.getAttribute(wmRuntime, "PendingRequests");

int stuckThreadCount = (Integer)myJMXWrapper.getAttribute(wmRuntime, "StuckThreadCount");

System.out.println(" Name: "+ name);

System.out.println(" Healthstate: "+ health);

System.out.println(" CompletedRequests: "+ completedRequests);

System.out.println(" PendingRequests: "+ pendingRequests);

System.out.println(" StuckThreadCount: "+ stuckThreadCount);

// get the different subcomponent

ObjectName myMaxThreadsConstraintRuntime =

(ObjectName)myJMXWrapper.getAttribute(wmRuntime, "MaxThreadsConstraintRuntime");

ObjectName myMinThreadsConstraintRuntime =

(ObjectName)myJMXWrapper.getAttribute(wmRuntime, "MinThreadsConstraintRuntime");

ObjectName myRequestClassRuntime =

(ObjectName)myJMXWrapper.getAttribute(wmRuntime, "RequestClassRuntime");

if (myMaxThreadsConstraintRuntime != null) {

String maxContraintName = (String)

myJMXWrapper.getAttribute(myMaxThreadsConstraintRuntime, "Name");

int maxContraintDeferredRequests = (Integer)

myJMXWrapper.getAttribute(myMaxThreadsConstraintRuntime, "DeferredRequests");

int maxContraintExecutingRequests = (Integer)

myJMXWrapper.getAttribute(myMaxThreadsConstraintRuntime, "ExecutingRequests");

System.out.println(" MaxThreadsConstraint: Name: "+maxContraintName+" DeferredRequests: "+

maxContraintDeferredRequests+" ExecutingRequests: "+

maxContraintExecutingRequests);

String maxONname=myMaxThreadsConstraintRuntime.getKeyProperty("Name");

if (maxONname != null) {

// try to get config mbean

Set<ObjectName> mySet = myJMXWrapper.getConnection().queryNames(

new ObjectName("\*:\*,Name="+maxONname+",Type=MaxThreadsConstraint"), null);

if (! mySet.isEmpty()) {

// get first

ObjectName myFirstConfigName = (ObjectName) mySet.iterator().next();

// get count

int maxCount = (Integer)myJMXWrapper.getAttribute(myFirstConfigName, "Count");

System.out.println(" -> max limit = "+maxCount);

}

else {

System.out.println(" -> no config mbean found !");

}

}

}

else {

System.out.println(" MaxThreadsConstraint not defined !");

}

if (myMinThreadsConstraintRuntime != null) {

String minContraintName = (String)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "Name");

long minContraintOutOfOrderExecutionCount = (Long)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "OutOfOrderExecutionCount");

int minContraintPendingRequests = (Integer)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "PendingRequests");

long minContraintCompletedRequests = (Long)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "CompletedRequests");

long minContraintMaxWaitTime = (Long)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "MaxWaitTime");

int minContraintMustRunCount = (Integer)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "MustRunCount");

long minContraintCurrentWaitTime = (Long)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "CurrentWaitTime");

int minContraintExecutingRequests = (Integer)

myJMXWrapper.getAttribute(myMinThreadsConstraintRuntime, "ExecutingRequests");

System.out.println(" MinThreadsConstraint: Name = "+ minContraintName);

System.out.println(" OutOfOrderExecutionCount = "

+ minContraintOutOfOrderExecutionCount);

System.out.println(" PendingRequests = "

+ minContraintPendingRequests);

System.out.println(" CompletedRequests = "

+ minContraintCompletedRequests);

System.out.println(" MaxWaitTime = " + minContraintMaxWaitTime);

System.out.println(" MustRunCount = " + minContraintMustRunCount);

System.out.println(" CurrentWaitTime = "

+ minContraintCurrentWaitTime);

System.out.println(" ExecutingRequests = "

+ minContraintExecutingRequests);

String minONname=myMinThreadsConstraintRuntime.getKeyProperty("Name");

if (minONname != null) {

// try to get config mbean

Set<ObjectName> mySet = myJMXWrapper.getConnection().queryNames(

new ObjectName("\*:\*,Name="+minONname+",Type=MinThreadsConstraint"), null);

if (! mySet.isEmpty()) {

// get first

ObjectName myFirstConfigName = (ObjectName) mySet.iterator().next();

// get count

int minCount = (Integer)myJMXWrapper.getAttribute(myFirstConfigName, "Count");

System.out.println(" -> min limit = "+minCount);

}

else {

System.out.println(" -> no config mbean found !");

}

}

}

else

System.out.println(" MinThreadsConstraint not defined !");

if (myRequestClassRuntime != null) {

long reqClassTotalThreadUse = (Long)

myJMXWrapper.getAttribute(myRequestClassRuntime, "TotalThreadUse");

double reqClassInterval = (Double)

myJMXWrapper.getAttribute(myRequestClassRuntime, "Interval");

long reqClassCompletedCount = (Long)

myJMXWrapper.getAttribute(myRequestClassRuntime, "CompletedCount");

long reqClassVirtualTimeIncrement = (Long)

myJMXWrapper.getAttribute(myRequestClassRuntime, "VirtualTimeIncrement");

long reqClassThreadUseSquares = (Long)

myJMXWrapper.getAttribute(myRequestClassRuntime, "ThreadUseSquares");

long reqClassDeltaFirst = (Long)myJMXWrapper.getAttribute(myRequestClassRuntime, "DeltaFirst");

int reqClassPendingRequestCount = (Integer)

myJMXWrapper.getAttribute(myRequestClassRuntime, "PendingRequestCount");

long reqClassDeltaRepeat = (Long)

myJMXWrapper.getAttribute(myRequestClassRuntime, "DeltaRepeat");

long reqClassMyLast = (Long)myJMXWrapper.getAttribute(myRequestClassRuntime, "MyLast");

String reqClassRequestClassType = (String)

myJMXWrapper.getAttribute(myRequestClassRuntime, "RequestClassType");

String reqClassName = (String)myJMXWrapper.getAttribute(myRequestClassRuntime, "Name");

System.out.println(" RequestClass: Name = "+reqClassName+

" is of type "+reqClassRequestClassType);

System.out.println(" TotalThreadUse = "+reqClassTotalThreadUse);

System.out.println(" Interval = "+reqClassInterval);

System.out.println(" CompletedCount = "+reqClassCompletedCount);

System.out.println(" VirtualTimeIncrement = "+reqClassVirtualTimeIncrement);

System.out.println(" ThreadUseSquares = "+reqClassThreadUseSquares);

System.out.println(" DeltaFirst = "+reqClassDeltaFirst);

System.out.println(" PendingRequestCount = "+reqClassPendingRequestCount);

System.out.println(" DeltaRepeat = "+reqClassDeltaRepeat);

System.out.println(" MyLast = "+reqClassMyLast);

}

else {

System.out.println(" RequestClass not defined !");

}

}

* query\_active\_transactions\_jmx

@SuppressWarnings("rawtypes")

public void printActiveJTATransactionValues() throws Exception {

ObjectName serverRuntime =

myJMXWrapper.getServerRuntime(myJMXWrapper.getMainServerDomainValues().get("serverName"));

ObjectName jtaRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JTARuntime");

// all active transactions

JTATransaction[] allActiveTransactions =

(JTATransaction[])myJMXWrapper.getAttribute(jtaRuntime, "JTATransactions");

//allActiveTransactions

if (allActiveTransactions != null)

for (int i=0;i<allActiveTransactions.length;i++) {

System.out.println("\nTransaction Nr.: "+

i+"\n========================================\n");

// transaction details

//The transaction ID assigned by the transaction manager.

try {

System.out.println("XID :"+

allActiveTransactions[i].getXid().getGlobalTransactionId().toString());

}

catch(Exception ex) {

System.out.println("XID : XID\_wrong\_or\_missing");

}

//The current status of the transaction.

System.out.println("Status :"+allActiveTransactions[i].getStatus());

//The URL of the coordinating server.

System.out.println("CoordinatorURL :"+allActiveTransactions[i].getCoordinatorURL());

// The name of the transaction as defined by the application that started the transaction.

System.out.println("Name :"+allActiveTransactions[i].getName());

//The time in seconds for which the transaction has been active.

System.out.println("SecondsActive :" +

allActiveTransactions[i].getSecondsActiveCurrentCount());

String[] txServers = allActiveTransactions[i].getServers();

if (txServers==null) txServers = new String[0];

//The servers that participate in the transaction.

StringBuffer txServerNames = new StringBuffer();

for (int s=0;s<txServers.length;s++)

txServerNames.append(txServers[s]+" ");

// add

System.out.println("Servers :"+txServerNames.toString());

// The collection of participating resource names and their status for the transaction.

Map mapResourceNamesAndStatus = allActiveTransactions[i].getResourceNamesAndStatus();

if (mapResourceNamesAndStatus!=null)

System.out.println("ResourceNamesAndStatus :"+printMap(mapResourceNamesAndStatus));

else

System.out.println("ResourceNamesAndStatus ----");

// The collection of participating servers and their status for the transaction.

Map mapServersAndStatus = allActiveTransactions[i].getServersAndStatus();

if (mapServersAndStatus!=null)

System.out.println("ServersAndStatus :"+printMap(mapServersAndStatus));

else

System.out.println("ServersAndStatus ----");

// The user-defined properties associated with the transaction.

Map mapUserProperties = allActiveTransactions[i].getUserProperties();

if (mapUserProperties!=null)

System.out.println("Properties :"+printMap(mapUserProperties));

else

System.out.println("Properties: ----");

}

}

* transactional\_resource\_statistic\_information\_jmx

public void printXAResourceInformationValues() throws Exception

{

ObjectName serverRuntime =

myJMXWrapper.getServerRuntime(myJMXWrapper.getMainServerDomainValues().get("serverName"));

ObjectName jtaRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JTARuntime");

// all XA Resources transactions

TransactionResourceRuntimeMBean[] allXAResources =

(TransactionResourceRuntimeMBean[])myJMXWrapper.getAttribute(jtaRuntime,

"TransactionResourceRuntimeMBeans");

//all XA resources

if (allXAResources != null)

for (int i=0;i<allXAResources.length;i++)

{

System.out.println("\nXA Resource Nr.: "+i);

// resource details

// The resource name

System.out.println("ResourceName: "+

myJMXWrapper.getAttribute(serverRuntime, "ResourceName"));

// The number of tx for which this resource has returned a heuristic commit decision

System.out.println("TransactionHeuristicCommitTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionHeuristicCommitTotalCount"));

// The number of tx for which this resource has reported a heuristic hazard decision

System.out.println("TransactionHeuristicHazardTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionHeuristicHazardTotalCount"));

// The number of transactions for which this resource has reported a heuristic mixed decision

System.out.println("TransactionHeuristicMixedTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionHeuristicMixedTotalCount"));

// The number of tx for which this resource has returned a heuristic rollback decision

System.out.println("TransactionHeuristicRollbackTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionHeuristicRollbackTotalCount"));

// The total number of transactions committed since the server was started.

System.out.println("TransactionCommittedTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionCommittedTotalCount"));

System.out.println("TransactionHeuristicsTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionHeuristicsTotalCount"));

// TransactionRolledBackTotalCount

System.out.println("TransactionRolledBackTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionRolledBackTotalCount"));

// The total number of transactions processed.

System.out.println("TransactionTotalCount: "+

myJMXWrapper.getAttribute(serverRuntime, "TransactionTotalCount"));

}

}

* monitor\_network\_channel\_jmx

public void printNetworkChannelInformation() throws Exception {

Hashtable<String,String> mainDomainValues = myJMXWrapper.getMainServerDomainValues();

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(mainDomainValues.get("serverName"));

// get all the different channel runtime mbeans

ObjectName[] serverChannelRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(serverRuntime, "ServerChannelRuntimes");

for (int i=0;i<serverChannelRuntimes.length;i++) {

String channelName = (String)myJMXWrapper.getAttribute(serverChannelRuntimes[i], "Name");

String channelURL = (String)myJMXWrapper.getAttribute(serverChannelRuntimes[i], "PublicURL");

long channelAcceptCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "AcceptCount");

long channelActiveConnectionsCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "ConnectionsCount");

long channelBytesReceivedCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "BytesReceivedCount");

long channelBytesSentCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "BytesSentCount");

long channelMessagesReceivedCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "MessagesReceivedCount");

long channelMessagesSentCount = (Long)

myJMXWrapper.getAttribute(serverChannelRuntimes[i], "MessagesSentCount");

System.out.println("Channelname: "+channelName+" (URL="+channelURL+")");

System.out.println(" Total connections count = "+channelAcceptCount);

System.out.println(" Actual connected connections = "

+channelActiveConnectionsCount);

System.out.println(" BytesReceived = "+channelBytesReceivedCount);

System.out.println(" BytesSent = "+channelBytesSentCount);

System.out.println(" MessagesReceived = "+channelMessagesReceivedCount);

System.out.println(" MessagesSent = "+channelMessagesSentCount);

System.out.println("\n");

}

}

* print\_datasource\_values\_jmx

public void printDatasourceInformation(String datasourceName, ObjectName jdbcRuntimeMBean)

throws Exception

{

// print Name

System.out.println("DATASOURCE: Information about the datasource: "+datasourceName);

System.out.println(" Enabled="+ myJMXWrapper.getAttribute(jdbcRuntimeMBean, "Enabled") +

" State="+ myJMXWrapper.getAttribute(jdbcRuntimeMBean, "State") );

// print ActiveConnectionsHighCount

System.out.println(" ActiveConnectionsHighCount="+

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "ActiveConnectionsHighCount"));

// print ActiveConnectionsCurrentCount

System.out.println(" ActiveConnectionsCurrentCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "ActiveConnectionsCurrentCount"));

// print ActiveConnectionsAverageCount

System.out.println(" ActiveConnectionsAverageCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "ActiveConnectionsAverageCount"));

// print ConnectionsTotalCount

System.out.println(" ConnectionsTotalCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "ConnectionsTotalCount"));

// print CurrCapacity

System.out.println(" CurrCapacity=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "CurrCapacity"));

// print CurrCapacityHighCount

System.out.println(" CurrCapacityHighCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "CurrCapacityHighCount"));

// print HighestNumAvailable

System.out.println(" HighestNumAvailable=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "HighestNumAvailable"));

// print LeakedConnectionCount

System.out.println(" LeakedConnectionCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "LeakedConnectionCount"));

// print FailuresToReconnectCount

System.out.println(" FailuresToReconnectCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "FailuresToReconnectCount"));

// print WaitSecondsHighCount

System.out.println(" WaitSecondsHighCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "WaitSecondsHighCount"));

// print WaitingForConnectionCurrentCount

System.out.println(" WaitingForConnectionCurrentCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "WaitingForConnectionCurrentCount"));

// print WaitingForConnectionFailureTotal

System.out.println(" WaitingForConnectionFailureTotal=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "WaitingForConnectionFailureTotal"));

// print WaitingForConnectionTotal

System.out.println(" WaitingForConnectionTotal=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "WaitingForConnectionTotal"));

// print WaitingForConnectionHighCount

System.out.println(" WaitingForConnectionHighCount=" +

myJMXWrapper.getAttribute(jdbcRuntimeMBean, "WaitingForConnectionHighCount"));

}

* print\_connections\_jmx

public void printConnectedJMSClients(String myServerName) throws WLSAutomationException

{

try

{

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(myServerName);

// get JMS runtime object reference

ObjectName myJMSRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JMSRuntime");

// get all JMS connections for all JMS servers for this runtime

// note that all connection instances are of type JMSConnectionRuntimeMBean

ObjectName[] jmsConnections = (ObjectName[])

myJMXWrapper.getAttribute(myJMSRuntime, "Connections");

// amount

System.out.println("Actually the "+myServerName+".jms runtime has " +

jmsConnections.length + " connections !");

// now examine all jms connections and get some basic informations

for (int i=0;i<jmsConnections.length;i++)

{

System.out.println("Connection: "+

myJMXWrapper.getAttribute(jmsConnections[i], "HostAddress")+

" with client id = "+myJMXWrapper.getAttribute(jmsConnections[i], "ClientID"));

// print all sessions associated with this connection

System.out.println(" has actually "+myJMXWrapper.getAttribute(jmsConnections[i],

"SessionsCurrentCount") +

" active sessions");

// get all sessions

ObjectName[] allConnectionJMSSessions = (ObjectName[])

myJMXWrapper.getAttribute(jmsConnections[i], "Sessions");

// iterate of sessions

for (int session=0;session<allConnectionJMSSessions.length;session++)

{

// print some session information

System.out.println(" Session: Active consumers: "+

myJMXWrapper.getAttribute(allConnectionJMSSessions[session], "ConsumersCurrentCount"));

System.out.println(" Active producer: "+

myJMXWrapper.getAttribute(allConnectionJMSSessions[session], "ProducersCurrentCount"));

System.out.println(" Messages: send:"+

myJMXWrapper.getAttribute(allConnectionJMSSessions[session], "MessagesSentCount")+

" received:"+myJMXWrapper.getAttribute(allConnectionJMSSessions[session],

"MessagesReceivedCount")+

" pending:"+myJMXWrapper.getAttribute(allConnectionJMSSessions[session],

"MessagesPendingCount"));

// iterate over producer

ObjectName[] activeJMSProducer = (ObjectName[])

myJMXWrapper.getAttribute(allConnectionJMSSessions[session], "Producers");

for(int producer=0;producer<activeJMSProducer.length;producer++)

{

// ... System.out.println(information you want -> see API);

}

// iterate over consumer

ObjectName[] activeJMSConsumers = (ObjectName[])

myJMXWrapper.getAttribute(allConnectionJMSSessions[session], "Consumers");

for(int consumer=0;consumer<activeJMSConsumers.length;consumer++)

{

// ... System.out.println(information you want -> see API);

}

}

}

}

catch(Exception ex)

{

throw new WLSAutomationException("Error in printJMSRuntimeState : "+ ex.getMessage());

}

}

* current\_session\_information\_jmx

public void printSessionPoolInformation(String myServerName, String myJMSServer) throws WLSAutomationException

{

try

{

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(myServerName);

// get JMS runtime object reference

ObjectName myJMSRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "JMSRuntime");

ObjectName myJMSServerReference = getListOfJMSServerFromRuntime(myJMSRuntime).get(myJMSServer);

if (myJMSServerReference==null)

{

System.out.println("JMS Server "+myJMSServer+" does not exit !");

}

else

{

System.out.println("The SessionPoolsCurrentCount of JMS server "+myJMSServer+" is "+

myJMXWrapper.getAttribute(myJMSRuntime, "SessionPoolsCurrentCount"));

System.out.println("The SessionPoolsHighCount of JMS server "+myJMSServer+" is "+

myJMXWrapper.getAttribute(myJMSRuntime, "SessionPoolsHighCount"));

System.out.println("The SessionPoolsTotalCount of JMS server "+myJMSServer+" is "+

myJMXWrapper.getAttribute(myJMSRuntime, "SessionPoolsTotalCount"));

}

}

catch(Exception ex)

{

throw new WLSAutomationException("Error in printJMSRuntimeState : "+ ex.getMessage());

}

}

* print\_states\_jmx

public void printQueueState(String myServerName, String myJMSServer, String myQueueName)

throws WLSAutomationException

{

try

{

ObjectName myQueue = getJMSDestinationRuntime(myQueueName, myJMSServer, myServerName);

if (myQueue != null)

{

System.out.println("Information about the queue: "+myQueueName);

// Detail states

System.out.println("The InsertionPausedState of queue "+myQueueName+" is "+

myJMXWrapper.getAttribute(myQueue, "InsertionPausedState"));

System.out.println("The ConsumptionPausedState of queue "+myQueueName+" is "+

myJMXWrapper.getAttribute(myQueue, "ConsumptionPausedState"));

System.out.println("The ProductionPausedState of queue "+myQueueName+" is "+

myJMXWrapper.getAttribute(myQueue, "ProductionPausedState"));

}

}

catch(Exception ex)

{

throw new WLSAutomationException("Error in printJMSRuntimeState : "+ ex.getMessage());

}

}

* message\_count\_jmx

public void printAmountOfMessagesInDestination(String myServerName,

String myJMSServer,

String myDestinationName)

throws WLSAutomationException

{

try

{

ObjectName myDestination = getJMSDestinationRuntime(myDestinationName,

myJMSServer, myServerName);

// System.out.println(amount of messages

System.out.println("The number of messages in the queue/topic "+myDestinationName +" is "+

myJMXWrapper.getAttribute(myDestination, "MessagesCurrentCount"));

// System.out.println(amount of pending messages

System.out.println("The number of messages in the queue/topic "+myDestinationName +" is "+

myJMXWrapper.getAttribute(myDestination, "MessagesPendingCount"));

}

catch(Exception ex)

{

throw new WLSAutomationException("Error in printJMSRuntimeState : "+ ex.getMessage());

}

}

* find\_application\_runtimes\_jmx

public HashMap<String, ObjectName> getApplicationNames(String serverName) throws Exception

{

HashMap<String, ObjectName> myResultList = new HashMap<String, ObjectName>();

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(serverName);

ObjectName[] allApplicationRuntimeMBeans = (ObjectName[])

myJMXWrapper.getAttribute(serverRuntime,"ApplicationRuntimes");

String nextApplicationName = null;

for (int i=0;i<allApplicationRuntimeMBeans.length;i++)

{

nextApplicationName = (String)

myJMXWrapper.getAttribute(allApplicationRuntimeMBeans[i],"Name");

Boolean isEAR = (Boolean) myJMXWrapper.getAttribute(allApplicationRuntimeMBeans[i],"EAR");

if (isEAR)

myResultList.put(nextApplicationName,allApplicationRuntimeMBeans[i]);

return myResultList;

}

}

* servlet\_values\_jmx

public void printServletInvocationCountInformation(ObjectName webAppComponentRuntime)

throws Exception

{

// calculcate the total summary of all calls

long invocationTotal = 0;

// get all servlet MBeans

ObjectName[] servletRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(webAppComponentRuntime, "Servlets");

for (int servletNumber=0; servletNumber<servletRuntimes.length; servletNumber++)

{

String nextName = (String)myJMXWrapper.getAttribute(servletRuntimes[servletNumber], "Name");

int contextInvocationTotalCount =

(Integer)myJMXWrapper.getAttribute(servletRuntimes[servletNumber], "InvocationTotalCount");

// sum up

invocationTotal += contextInvocationTotalCount;

// print to list

System.out.println(" Servlet:"+nextName+" Invocations:"+

new Integer(contextInvocationTotalCount));

}

// print SUMMARY to list

System.out.println(" Overall-InvocationTotalCount" + new Long(invocationTotal));

}

* EJB\_print\_information\_jmx

public HashMap<String, ObjectName> getEJBComponentNames(ObjectName applicationRuntimeMBean)

throws Exception

{

HashMap<String, ObjectName> myResultList = new HashMap<String, ObjectName>();

// get component runtimes

ObjectName[] componentRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(applicationRuntimeMBean, "ComponentRuntimes");

for (int compNumber=0; compNumber < componentRuntimes.length; compNumber++)

{

String componentType = (String)

myJMXWrapper.getAttribute(componentRuntimes[compNumber], "Type");

if (componentType.toString().equals("EJBComponentRuntime"))

{

String name = (String) myJMXWrapper.getAttribute(componentRuntimes[compNumber], "Name");

myResultList.put(name,componentRuntimes[compNumber]);

}

}

return myResultList;

}

* EJB\_print\_runtime\_jmx

public void printEJBInformation(ObjectName ejbComponentRuntime, boolean printDetails)

throws Exception

{

String myName = (String)myJMXWrapper.getAttribute(ejbComponentRuntime, "Name");

System.out.println("Found EJB modul "+myName+" with current deployment state = " +

getApplicationComponentState(ejbComponentRuntime));

// now look at the submodules

ObjectName[] myEJBs = (ObjectName[])myJMXWrapper.getAttribute(ejbComponentRuntime, "EJBRuntimes");

for (int i=0;i<myEJBs.length;i++)

{

String ejbName = (String)myJMXWrapper.getAttribute(myEJBs[i], "Name");

String ejbType = (String)myJMXWrapper.getAttribute(myEJBs[i], "Type");

System.out.println(" EJB: "+ejbName+" is of type "+ejbType);

if (printDetails)

{

// all have transactions

//transaction information from TransactionRuntime

ObjectName myTxRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "TransactionRuntime");

System.out.println(" TransactionInfo: total:"+

myJMXWrapper.getAttribute(myTxRuntime, "TransactionsCommittedTotalCount")+

"rolledback:"+myJMXWrapper.getAttribute(myTxRuntime,

"TransactionsRolledBackTotalCount")+

" timedout:" +myJMXWrapper.getAttribute(myTxRuntime,

"TransactionsTimedOutTotalCount"));

if ("StatelessEJBRuntime".equals(ejbType)) {

// pool information from PoolRuntime

ObjectName myPoolRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "PoolRuntime");

System.out.println(" PoolInfo: totalAccess:"+

myJMXWrapper.getAttribute(myPoolRuntime, "AccessTotalCount")+

" beansInUse:"+myJMXWrapper.getAttribute(myPoolRuntime, "BeansInUseCount")+

" beansInUseCurrent:"+myJMXWrapper.getAttribute(myPoolRuntime,

"BeansInUseCurrentCount")+

" destroyed:"+myJMXWrapper.getAttribute(myPoolRuntime, "DestroyedTotalCount")+

" idle:"+myJMXWrapper.getAttribute(myPoolRuntime, "IdleBeansCount")+

" pooledCurrent:"+myJMXWrapper.getAttribute(myPoolRuntime,

"PooledBeansCurrentCount")+

" timedout:"+myJMXWrapper.getAttribute(myPoolRuntime, "TimeoutTotalCount"));

// timer consists of a list of timers

ObjectName myTimerRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "TimerRuntime");

if (myTimerRuntime != null)

System.out.println(" Timer information: Name:"+

myJMXWrapper.getAttribute(myTimerRuntime, "Name")+

" activeTimers:"+myJMXWrapper.getAttribute(myTimerRuntime, "ActiveTimerCount")+

" timeout:"+myJMXWrapper.getAttribute(myTimerRuntime, "TimeoutCount")+

" cancelled:"+myJMXWrapper.getAttribute(myTimerRuntime, "CancelledTimerCount")+

" disabled:"+myJMXWrapper.getAttribute(myTimerRuntime, "DisabledTimerCount"));

}

else if ("StatefulEJBRuntime".equals(ejbType)) {

// cache information from CacheRuntime

ObjectName myCacheRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "CacheRuntime");

System.out.println(" CacheInfo: hits:"+

myJMXWrapper.getAttribute(myCacheRuntime, "CacheHitCount")+

" currentBeans:"+myJMXWrapper.getAttribute(myCacheRuntime,

"CachedBeansCurrentCount")+

" access:"+myJMXWrapper.getAttribute(myCacheRuntime, "CacheAccessCount"));

// locking information from LockingRuntime

ObjectName myLockingRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "LockingRuntime");

System.out.println(" LockingInfo: currentCount:"+

myJMXWrapper.getAttribute(myLockingRuntime, "LockEntriesCurrentCount")+

" accessCount:"+myJMXWrapper.getAttribute(myLockingRuntime, "LockManagerAccessCount")+

" timeoutTotalCount:"+myJMXWrapper.getAttribute(myLockingRuntime,

"TimeoutTotalCount"));

}

else if ("EntityEJBRuntime".equals(ejbType))

{

ObjectName entityPool = (ObjectName)myJMXWrapper.getAttribute(myEJBs[i], "PoolRuntime");

System.out.println(" PooledBeansCurrent ="+

myJMXWrapper.getAttribute(entityPool,"PooledBeansCurrentCount")+

" AccessTotal ="+ myJMXWrapper.getAttribute(entityPool,"AccessTotalCount")+

" DestroyedTotal ="+ myJMXWrapper.getAttribute(entityPool,"DestroyedTotalCount")+

" IdleBeans ="+ myJMXWrapper.getAttribute(entityPool,"IdleBeansCount")+

" BeansInUse ="+ myJMXWrapper.getAttribute(entityPool,"BeansInUseCount")+

" BeansInUseCurrent ="+

myJMXWrapper.getAttribute(entityPool,"BeansInUseCurrentCount")+

" WaiterTotal ="+ myJMXWrapper.getAttribute(entityPool,"WaiterTotalCount")+

" WaiterCurrent ="+ myJMXWrapper.getAttribute(entityPool,"WaiterCurrentCount")+

" TimeoutTotal ="+ myJMXWrapper.getAttribute(entityPool,"TimeoutTotalCount"));

}

else if ("MessageDrivenEJBRuntime".equals(ejbType)) {

// print the mdb status

System.out.println(" ConnectionStatus = "+

myJMXWrapper.getAttribute(myEJBs[i], "ConnectionStatus");

// print status of the MDB

System.out.println(" MDBStatus = "+myJMXWrapper.getAttribute(myEJBs[i], "MDBStatus");

// client id

System.out.println(" JmsClientID = "+myJMXWrapper.getAttribute(myEJBs[i], "JmsClientID");

// count of processed messages

System.out.println(" ProcessedMessageCount = "+

myJMXWrapper.getAttribute(myEJBs[i], "ProcessedMessageCount");

// amout of suspended messages

System.out.println(" SuspendCount = "+

myJMXWrapper.getAttribute(myEJBs[i], "SuspendCount");

// healthstate of MDB

System.out.println(" HealthState="+getHealthStateInformation((HealthState)

myJMXWrapper.getAttribute(myEJBs[i],"HealthState")));

// pool information from PoolRuntime

ObjectName myPoolRuntime = (ObjectName)myJMXWrapper.getAttribute(myEJBs[i], "PoolRuntime");

System.out.println(" PoolInfo: totalAccess:"+

myJMXWrapper.getAttribute(myPoolRuntime, "AccessTotalCount")+

" beansInUse:"+myJMXWrapper.getAttribute(myPoolRuntime, "BeansInUseCount")+

" beansInUseCurrent:"+myJMXWrapper.getAttribute(myPoolRuntime,

"BeansInUseCurrentCount")+

" destroyed:"+myJMXWrapper.getAttribute(myPoolRuntime, "DestroyedTotalCount")+

" idle:"+myJMXWrapper.getAttribute(myPoolRuntime, "IdleBeansCount")+

" pooledCurrent:"+myJMXWrapper.getAttribute(myPoolRuntime,

"PooledBeansCurrentCount")+

" timedout:"+myJMXWrapper.getAttribute(myPoolRuntime, "TimeoutTotalCount"));

// timer consists of a list of timers

ObjectName myTimerRuntime = (ObjectName)

myJMXWrapper.getAttribute(myEJBs[i], "TimerRuntime");

if (myTimerRuntime != null)

System.out.println(" Timer information: Name:"+

myJMXWrapper.getAttribute(myTimerRuntime, "Name")+

" activeTimers:"+myJMXWrapper.getAttribute(myTimerRuntime, "ActiveTimerCount")+

" timeout:"+myJMXWrapper.getAttribute(myTimerRuntime, "TimeoutCount")+

" cancelled:"+myJMXWrapper.getAttribute(myTimerRuntime, "CancelledTimerCount")+

" disabled:"+myJMXWrapper.getAttribute(myTimerRuntime, "DisabledTimerCount"));

}

}

}

}

* get\_JCA\_component\_list

public HashMap<String, ObjectName> getJCAConnectorRuntimes(ObjectName applicationRuntimeMBean)

throws Exception

{

HashMap<String, ObjectName> myResultList = new HashMap<String, ObjectName>();

// get component runtimes

ObjectName[] componentRuntimes = (ObjectName[])

myJMXWrapper.getAttribute(applicationRuntimeMBean, "ComponentRuntimes");

for (int compNumber=0; compNumber < componentRuntimes.length; compNumber++)

{

String componentType =

(String)myJMXWrapper.getAttribute(componentRuntimes[compNumber], "Type");

if (componentType.toString().equals("ConnectorComponentRuntime"))

{

String name = (String) myJMXWrapper.getAttribute(componentRuntimes[compNumber], "Name");

myResultList.put(name,componentRuntimes[compNumber]);

}

}

return myResultList;

}

* print\_JCA\_connector\_values

public void printConnectorMonitoringValues(ObjectName connectorRuntime) throws Exception

{

String myName = (String)myJMXWrapper.getAttribute(connectorRuntime, "Name");

System.out.println("Found Connector modul "+myName+

" with current deployment state = " + getApplicationComponentState(connectorRuntime));

int myDeploymentState = (Integer) myJMXWrapper.getAttribute(connectorRuntime,"DeploymentState");

String d\_state = "UNKNOWN";

if (myDeploymentState == weblogic.management.runtime.ComponentRuntimeMBean.UNPREPARED)

d\_state = "UNPREPARED";

else if(myDeploymentState == weblogic.management.runtime.ComponentRuntimeMBean.PREPARED)

d\_state = "PREPARED";

else if(myDeploymentState == weblogic.management.runtime.ComponentRuntimeMBean.ACTIVATED)

d\_state = "ACTIVATED";

else if(myDeploymentState == weblogic.management.runtime.ComponentRuntimeMBean.NEW)

d\_state = "NEW";

// add T\_DeploymentState

System.out.println(" DeploymentState: "+ d\_state);

ObjectName[] myConPools = (ObjectName[])

myJMXWrapper.getAttribute(connectorRuntime,"ConnectionPools");

if (myConPools.length > 0)

{

// monitor only first pool !

System.out.println(" ActiveConCurrent: "+

myJMXWrapper.getAttribute(connectorRuntime, "ActiveConnectionsCurrentCount"));

System.out.println(" ActiveConHigh: "+

myJMXWrapper.getAttribute(connectorRuntime, "ActiveConnectionsHighCount"));

System.out.println(" ConCreatedTotal: "+

myJMXWrapper.getAttribute(connectorRuntime, "ConnectionsCreatedTotalCount"));

System.out.println(" ConRejectedTotal: "+

myJMXWrapper.getAttribute(connectorRuntime, "ConnectionsRejectedTotalCount"));

System.out.println(" ConDestroyedTotal: "+

myJMXWrapper.getAttribute(connectorRuntime, "ConnectionsDestroyedTotalCount"));

System.out.println(" FreeConCurrent: "+

myJMXWrapper.getAttribute(connectorRuntime, "FreeConnectionsCurrentCount"));

System.out.println(" FreeConHigh: "+

myJMXWrapper.getAttribute(connectorRuntime, "FreeConnectionsHighCount"));

System.out.println(" AverageActiveUsage: "+

myJMXWrapper.getAttribute(connectorRuntime, "AverageActiveUsage"));

System.out.println(" CloseCount: "+

myJMXWrapper.getAttribute(connectorRuntime, "CloseCount"));

System.out.println(" ConnectionsDestroyedByErrorTotalCount: "+

myJMXWrapper.getAttribute(connectorRuntime, "ConnectionsDestroyedByErrorTotalCount"));

System.out.println(" ConnectionsDestroyedByShrinkingTotalCount: "+

myJMXWrapper.getAttribute(connectorRuntime,

"ConnectionsDestroyedByShrinkingTotalCount"));

System.out.println(" ConnectionsMatchedTotalCount: "+

myJMXWrapper.getAttribute(connectorRuntime,

"ConnectionsMatchedTotalCount"));

System.out.println(" CurrentCapacity: "+

myJMXWrapper.getAttribute(connectorRuntime, "CurrentCapacity"));

System.out.println(" MaxCapacity: "+

myJMXWrapper.getAttribute(connectorRuntime, "MaxCapacity"));

System.out.println(" MaxIdleTime: "+

myJMXWrapper.getAttribute(connectorRuntime, "MaxIdleTime"));

System.out.println(" NumUnavailableCurrentCount: "+

myJMXWrapper.getAttribute(connectorRuntime, "NumUnavailableCurrentCount"));

System.out.println(" NumUnavailableHighCount: "+

myJMXWrapper.getAttribute(connectorRuntime, "NumUnavailableHighCount"));

System.out.println(" NumWaiters: "+

myJMXWrapper.getAttribute(connectorRuntime, "NumWaiters"));

System.out.println(" NumWaitersCurrentCount: "+

myJMXWrapper.getAttribute(connectorRuntime, "NumWaitersCurrentCount"));

System.out.println(" RecycledTotal: "+

myJMXWrapper.getAttribute(connectorRuntime, "RecycledTotal"));

System.out.println(" ShrinkCountDownTime: "+

myJMXWrapper.getAttribute(connectorRuntime, "ShrinkCountDownTime"));

System.out.println(" ShrinkPeriodMinutes: "+

myJMXWrapper.getAttribute(connectorRuntime, "ShrinkPeriodMinutes"));

}

}

* monitoring\_values\_jmx

public void monitorWTC() throws Exception {

// get main values

Hashtable<String,String> mainValues = myJMXWrapper.getMainServerDomainValues();

// go to MBean

ObjectName serverRuntime = myJMXWrapper.getServerRuntime(mainValues.get("serverName"));

ObjectName wtcRuntime = (ObjectName) myJMXWrapper.getAttribute(serverRuntime, "WTCRuntime");

// list connections configured

DSessConnInfo[] myWTCConnectionInfos = (DSessConnInfo[])

myJMXWrapper.invoke(wtcRuntime,"listConnectionsConfigured",

new Object[]{},new String[]{});

System.out.println("DSessConnInfo:");

for (int i=0;i<myWTCConnectionInfos.length;i++) {

DSessConnInfo nextDSessConnInfo = myWTCConnectionInfos[i];

System.out.println("DSessConnInfo-"+i);

System.out.println(" "+nextDSessConnInfo.getLocalAccessPointId());

System.out.println(" "+nextDSessConnInfo.getRemoteAccessPointId());

System.out.println(" "+nextDSessConnInfo.getConnected());

}

// get all service information

DServiceInfo[] allDServiceInfos = (DServiceInfo[])

myJMXWrapper.getAttribute(wtcRuntime, "ServiceStatus");

System.out.println("\n\nDServiceInfos:");

for (int i=0;i<allDServiceInfos.length;i++) {

System.out.println("DSessConnInfo-"+i);

System.out.println(" "+allDServiceInfos[i].getLocalAccessPoint());

System.out.println(" "+allDServiceInfos[i].getServiceName());

System.out.println(" "+

WTCServiceStatus.svcTypeToString(allDServiceInfos[i].getServiceType()));

System.out.println(" "+WTCServiceStatus.statusToString(allDServiceInfos[i].getStatus()));

}

}

* interface\_definition\_jmx

public interface PushExampleMBean

{

/\*\*

\* Is plugin enabled - means does plugin record periodically the runtime values and report them

\* @return String

\*/

public String getEnabled();

/\*\*

\* VERY IMPORTANT: Activates or deactivates the gathering and pushing

\* @param enabled String

\*/

public void setEnabled(String enabled);

/\*\*

\* Interval in seconds how often the information are collected and reported

\* @return interval

\*/

public int getUpdateIntervalInSeconds();

/\*\*

\* Set the interval how often the runtime information should be gathered and reported

\* @param intervalInSeconds int

\*/

public void setUpdateIntervalInSeconds(int intervalInSeconds);

/\*\*

\* GEt the destination URL:

\* @return the destination URL

\*/

public String getDestinationURL();

/\*\*

\* Set the destination URL:

\*/

public void setDestinationURL(String destinationURL);

}

* implement\_constructor\_jmx

public class PushExampleMBeanImpl extends StandardMBean implements PushExampleMBean

{

...

public PushExampleMBeanImpl(JMXWrapperLocal \_JMXWrapper) throws Exception {

super(PushExampleMBean.class, false);

try

{

System.out.println("WLSMonitoringMBean - Init called");

myJMXWrapper = \_JMXWrapper;

myJMXWrapper.connectToAdminServer(false,null,null,null);

mainDomainValues = myJMXWrapper.getMainServerDomainValues();

ObjectName mymbean = new

ObjectName("pushMonitoring:Name=ExamplePushInformation,Type=ExamplePushPlugin");

// now register in MBeanServer

((MBeanServer)myJMXWrapper.getConnection()).registerMBean(this, mymbean);

System.out.println("Push-Monitoring: PushExampleMBean - MBean registered");

// if running on adminserver -> change to domainruntime

String isCurrentServerAnAdminServer = mainDomainValues.get("connectedToAdminServer");

if ("true".equalsIgnoreCase(isCurrentServerAnAdminServer)) // is Admin

{

myJMXWrapper.disconnectFromAdminServer();

myJMXWrapper.connectToAdminServer(true,null,null,null);

connectedToAdmin = true;

System.out.println("Monitoring: Connection changed to DomainRuntime as “+

”THIS is the admin server !");

}

// example implementation in order to start data thread

setEnabled(getEnabled());

System.out.println("PushExample-Monitoring: init completed");

}

catch(Exception ex) {

ex.printStackTrace();

throw ex;

}

}

* plug-in\_check\_jmx

/\*\*

\* Is plugin enabled - means does plugin record periodically the runtime values and report them

\* @return boolean

\*/

public String getEnabled()

{

return pluginEnabled ? "true" : "false" ;

}

public void setEnabled(String newEnabledValue)

{

// set value

pluginEnabled = Boolean.parseBoolean(newEnabledValue);

if (pluginEnabled==true) // ENABLE

{

if (myPushExampleThread != null)

{

System.out.println("Monitoring is already active !!");

}

else

{

System.out.println("Monitoring will be started !!");

myPushExampleThread = new PushExampleThread(this);

myPushExampleThread.setDaemon(true);

myPushExampleThread.start();

}

}

else // DISABLE

{

if (myPushExampleThread == null)

{

System.out.println("Monitoring is already disabled !!");

}

else

{

// setting enabled to false is enough - will be picked up by thread

System.out.println("Monitoring will be disabled !!");

myPushExampleThread.interrupt();

myPushExampleThread = null;

}

}

}

* push\_data\_jmx

public class PushExampleThread extends Thread

{

private PushExampleMBeanImpl myMBean = null;

private PushExampleDestinations myPushExampleDestination= null;

public PushExampleThread(PushExampleMBeanImpl \_myMBean) {

myMBean = \_myMBean;

myPushExampleDestination = new PushExampleDestinations(myMBean);

}

public void run()

{

try

{

// do pushing while plugin is enabled

while (myMBean.getEnabled().equalsIgnoreCase("true"))

{

int waitSeconds = myMBean.getUpdateIntervalInSeconds();

// wait

try {

Thread.sleep(waitSeconds\*1000);

}

catch (Exception ex) {

continue;

}

// still enabled ?

if (myMBean.getEnabled().equalsIgnoreCase("true"))

{

// get monitoring values and push to destination

// if admin => push domain data

if (myMBean.isConnectedToAdmin())

myPushExampleDestination.pushData("DOMAIN", myMBean.getDomainData());

// push server data

myPushExampleDestination.pushData("SERVER", myMBean.getServerData());

}

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

}

* get\_domain\_data\_jmx\_(partial)

public HashMap<String, String> getDomainData() throws Exception

{

HashMap<String, String> myResultList = new HashMap<String, String>();

// add domain name

myResultList.put("DomainName", (String)mainDomainValues.get("domainName"));

ArrayList<String> msServerNames = myJMXWrapper.getManagedServerNames();

int amountOfMSServer = msServerNames.size();

int amountRunning = 0;

int amountShutdown = 0;

int amountFailedAdmin = 0;

// iterator over server list and get server info

for (int i=0; i<amountOfMSServer;i++)

{

String nextState = myJMXWrapper.getServerState(msServerNames.get(i));

if ("RUNNING".equalsIgnoreCase(nextState))

amountRunning++;

else if ("SHUTDOWN".equalsIgnoreCase(nextState) ||

"SHUTTING\_DOWN".equalsIgnoreCase(nextState))

amountShutdown++;

if ("UNKNOWN".equalsIgnoreCase(nextState) ||

"FAILED\_NOT\_RESTARTABLE".equalsIgnoreCase(nextState) ||

"FAILED".equalsIgnoreCase(nextState) ||

"FAILED\_RESTARTING".equalsIgnoreCase(nextState) ||

"FORCE\_SHUTTING\_DOWN".equalsIgnoreCase(nextState)

)

amountFailedAdmin++;

}

// looking only at first cluster for demonstration reasons

// e.g.: com.bea:Name=TestDomain,Type=Domain

ObjectName myDomainMBean = new ObjectName("com.bea:Name=" +

(String)mainDomainValues.get("domainName") +",Type=Domain");

ObjectName[] allClusters = (ObjectName[])myJMXWrapper.getAttribute(myDomainMBean,"Clusters");

if (allClusters.length > 0)

{

// use 1st cluster !!!

// get cluster member

ObjectName[] clusterServer =

(ObjectName[])myJMXWrapper.getAttribute(allClusters[0],"Servers");

int amountClusterServerRunning = 0;

int amountClusterServerShutdown = 0;

int amountClusterServerNotWorkingOrFailed = 0;

// iterate over cluster member

for (int i=0; i<clusterServer.length;i++)

{

String nextState = myJMXWrapper.getServerState(

(String)myJMXWrapper.getAttribute(clusterServer[i],"Name"));

if ("RUNNING".equalsIgnoreCase(nextState))

amountClusterServerRunning++;

else if ("SHUTDOWN".equalsIgnoreCase(nextState))

amountClusterServerShutdown++;

else // all other states

amountClusterServerNotWorkingOrFailed++;

}

// NOTE: In this example: Only looking at 1st cluster

if (clusterServer.length == amountClusterServerRunning) // all running

myResultList.put("ClusterState", "RUNNING");

else if (clusterServer.length == amountClusterServerShutdown)

// all shutdown or failed :-((

myResultList.put("ClusterState", "SHUTDOW");

else if (clusterServer.length == amountClusterServerNotWorkingOrFailed)

// all shutdown or failed :-((

myResultList.put("ClusterState", "NOT WORKING OR FAILED");

else if (amountClusterServerRunning>0) // partly running

myResultList.put("ClusterState", "PARTLY RUNNING");

else

myResultList.put("ClusterState", "NOT WORKING OR FAILED");

}

else

{

// no cluster ?!?!?!

myResultList.put("ClusterState", "NO Cluster configured");

}

// add values to result list

myResultList.put("NrServers", ""+amountOfMSServer);

myResultList.put("SrvRunning", ""+amountRunning);

myResultList.put("SrvFailedAdmin", ""+amountFailedAdmin);

myResultList.put("SrvDown", ""+amountShutdown);

return myResultList;

}

* push\_data\_jmx\_(partial)

public class PushExampleDestinations

{

// this is only a dummy implementation as it does nothing but print the values to system.out.

// depending on the supported destinations (e.g. http, file, corba(iiop), webservice

// appropriate endpoint communication must be implemented here

private PushExampleMBeanImpl myPushExampleMBeanImpl = null;

public PushExampleDestinations(PushExampleMBeanImpl myBean) {

myPushExampleMBeanImpl = myBean;

// get connection URL an initialize appropriate backend connectivity

// this should be outsourced into own classes according to the well known design patterns

// e.g. String connURI = myPushExampleMBeanImpl.getDestinationURL();

// if (connURI.startswith('http://') .... http backend

// else if (connURI.startswith("iiop://") .... corba backend

// ...

}

// only simple example to write out data

public void pushData(String dataCategory, HashMap<String, String> data) {

System.out.println("\nMonitoring Data - Push simulation:\n===============\n");

System.out.println(" Category: " + dataCategory);

Iterator<String> it = data.keySet().iterator();

while (it.hasNext()) {

String name = it.next();

String value = data.get(name);

System.out.println(" "+name+" = "+value);

}

}

}

* set\_debug\_flags\_jmx

public void setDebugFlags(String serverName, Properties debugProps) throws WLSAutomationException

{

try {

// get the server runtime(!)

ObjectName myServer = (ObjectName)myJMXWrapper.invoke(myJMXWrapper.getDomainConfigRoot(),

"lookupServer",

new Object[]{new String(serverName)},

new String[]{String.class.getName()});

// get the ServerDebug mbean

ObjectName myServerDebugMBean = (ObjectName)myJMXWrapper.getAttribute(myServer, "ServerDebug");

// iterate of debugProps and set them !

Iterator it = debugProps.keySet().iterator();

while (it.hasNext())

{

String nextkey = (String)it.next();

String nextvalue = debugProps.getProperty(nextkey);

// set debug value

myJMXWrapper.setAttribute(myServerDebugMBean, new Attribute(nextkey,new Boolean(nextvalue)));

System.out.println("Setting "+nextkey+" to value "+nextvalue);

}

}

catch(Exception ex) {

throw new WLSAutomationException("Error while getThreadDump of server "+serverName+" : "+ ex.getMessage());

}

}

* configure\_JPS\_keystore

String operation\_name = "addKeyStoreService";

ObjectName myObjectName = new ObjectName(com.oracle.jps:type=JpsConfig);

// Example values

String param\_path = "/opt/keystores/oswm/test.jks";

String param\_typ = "JKS";

String param\_pwd = "myKeystorePassword";

String param\_signAlias = "mySignAlias";

String param\_signPwd = "myPwd";

String param\_encAlias = "myEncryptionAlias";

String param\_encPwd = "myEncrPassword";

// try to remove old service KEYSTORE. This has proven to be necessary in case another keystore

// already exists. Reason unknown

try {

myJMXWrapper.invoke(myObjectName,"removeServiceType",

new Object[] { null, "KEY\_STORE" },

new String[] { String.class.getName(), String.class.getName() });

}

catch (Exception ex) {

// Log but ignore exception

System.out.println("Exception "+ex.getMessage()+" while deleting old keystore -> can be IGNORED !");

}

HashMap<String,String> myMap = new HashMap<String,String>();

myMap.put("keystore.csf.map", "oracle.wsm.security");

PortableMap myOracleMap = new PortableMap(myMap);

//create keystore

System.out.println("Creating Keystore !");

myJMXWrapper.invoke(myObjectName, operation\_name,

new Object[] { null,

param\_path,

param\_typ,

param\_pwd.toCharArray(),

param\_signAlias,

param\_signPwd.toCharArray(),

param\_encAlias,

param\_encPwd.toCharArray(),

myOracleMap.toCompositeData(PortableMap.toCompositeType())

},

new String[] { String.class.getName(),

String.class.getName(),

String.class.getName(),

"[C",

String.class.getName(),

"[C",

String.class.getName(),

"[C",

"javax.management.openmbean.CompositeData"

});

System.out.println("Keystore configuration has been created !");

* define\_issuer\_URI

System.out.println("Setup Issuer URIs");

String[] myURIs = new String[]{"http://services.wlst\_automation.de/wlstbook/MyPartner"};

myJMXWrapper.invoke(

new ObjectName("com.oracle.jps:type=JpsConfig"),

"updateSAMLLoginModule",

new Object[] { "SAML", // Context

"saml.loginmodule", // name of login module

"required", // ControlFlag

new Boolean(true), // set debug to true

new Boolean(true), // add roles

null,

myURIs)

},

new String[] { String.class.getName(),

String.class.getName(),

String.class.getName(),

Boolean.class.getName(),

Boolean.class.getName(),

javax.management.openmbean.CompositeData,

"[Ljava.lang.String;"

});

* OWSM\_issuer

String issuer\_uri = "http://services.wlst\_automation.de/wlstbook/ThatIsMe"

ObjectName myObjectName = new ObjectName("oracle.j2ee.config:name=pap,type=PolicyAccessorConfig");

String parameter\_name ="SamlSvTrustedDns";

String value ="wlst\_automation@mh-enterpriseconsulting.de, CN=WLST, OU=PRODUCTION, O=MH-EnterpriseConsulting, C=DE";

// Build openmbean structure

String[] itemNames = new String[]{"key", "value"};

CompositeType rowInTabular = new CompositeType("java.util.Map<java.lang.String, java.util.List<java.lang.String>>","java.util.Map<java.lang.String, java.util.List<java.lang.String>>",

itemNames,itemNames,

new OpenType[]{SimpleType.STRING, new ArrayType(1,SimpleType.STRING)} );

TabularType myMapTabularType = new TabularType("IssuerURIs", "List of IssuerURIs", rowInTabular, new String[]{"key"});

TabularDataSupport tabularSupport = null;

// try to get from server, if null create new

tabularSupport = (TabularDataSupport)myJMXWrapper.getAttribute(myObjectName,parameter\_name);

if (tabularSupport == null)

tabularSupport = new TabularDataSupport(myMapTabularType);

// compile data

CompositeData compositeData = new CompositeDataSupport(rowInTabular,

itemNames,

new Object[]{issuer\_uri, new String[]{value}});

tabularSupport.put(compositeData);

// set Attribute

myJMXWrapper.setAttribute(myObjectName,new Attribute(parameter\_name, tabularSupport));

* OWSM\_user\_configuration

String key\_value = "myOWSMaccesskey";

String alias = "oracle.wsm.security";

String user = "MyOWSMaccessUser";

String password = "lfhajfh3297562935jdhso325z";

ObjectName myObjectName = new ObjectName("com.oracle.jps:type=JpsCredentialStore");

// Use an WebLogic class to create th correct credentials

// package to be used: oracle.security.jps.mas.mgmt.jmx.credstore

PortableCredential cred = new PortablePasswordCredential(user,password.toCharArray(),"OWSM Access UserInfo");

// key and alias exists ?

boolean alreadyExists = (Boolean) myJMXWrapper.invoke(myObjectName,"containsCredential",

new Object[] { alias, key\_value },

new String[] { String.class.getName(), String.class.getName() });

// check if alias and key exists and if yes, delete it

if (alreadyExists) {

myJMXWrapper.invoke(myObjectName,"deleteCredential",

new Object[] { alias, key\_value },

new String[] { String.class.getName(), String.class.getName() });

}

//create a password credential

myJMXWrapper.invoke(myObjectName, operation\_name,

new Object[] { alias, key\_value, cred.toCompositeData(null) },

new String[] { String.class.getName(), String.class.getName(),CompositeData.class.getName() });

System.out.println("Alias="+alias+" , Key="+key\_value+" has been created !");

* export\_policy\_sets

ObjectName myObjectName = new

ObjectName("oracle.wsm:Location=AdminServer,name=WSMDocumentManager,type=Repository");

String export\_directory = "/data/owsm/export";

StringBuffer buf = new StringBuffer();

System.out.println("Export all Policy Sets");

IDocumentManager idm = JMX.newMXBeanProxy(myJMXWrapper.getConnection(), myObjectName, IDocumentManager.class);

RepositoryCommands wsm\_rcs = new RepositoryCommands(idm);

String retString = wsm\_rcs.beginRepositorySession();

// Step 1: Export all policy sets !

List<String> ps\_names = wsm\_rcs.listPolicySets(null);

for (int i=0;i<ps\_names.size();i++)

{

String psName = ps\_names.get(i).trim();

// ignore this entry.

if (! "Global Policy Sets in Repository:".equals(psName)) {

buf.append("POLICYSET-Name = "+psName+"\n==========================================\n");

List<String> polSet = wsm\_rcs.displayPolicySet(psName);

for (int ps = 0; ps < polSet.size(); ps++)

buf.append(" " + polSet.get(ps)+"\n");

}

}

System.out.println("END - List all Policy Sets");

// abort session. No need to commit as we did not change anything

wsm\_rcs.abortRepositorySession();

// write in File

FileUtils.writeFile(export\_directory+"/\_all\_policy\_sets.txt",buf.toString());

* export\_single\_policies

String[] signature = new String[] {"java.lang.String"}; // mode = "WRITE\_ALWAYS"

Object[] params = new Object[]{ null};

String[] nameList = (String[])myJMXWrapper.invoke(myObjectName, "retrieveDocumentNames",

params, signature);

ArrayList<String> myPolicyDocs = new ArrayList<String>();

for (int i=0;i<nameList.length;i++)

if (nameList[i].startsWith("/policies/")) {

System.out.println"Found policy: "+nameList[i]);

myPolicyDocs.add(nameList[i]);

}

// now construct the search string

String searchString = null;

if (myPolicyDocs.size() != 0) {

searchString = "|";

for (int i=0;i<wespeDocs.size();i++)

searchString=searchString+"("+ myPolicyDocs.get(i)+")";

}

// now export the policies

String[] signatureExport = new String[]

{"java.lang.String","javax.management.openmbean.TabularData"};

Object[] paramsExport = new Object[]{ searchString, null};

// read the policies

Map myPolicies = (Map)myJMXWrapper.invoke(myObjectName, "retrieveDocuments", paramsExport,

signatureExport);

Iterator it = myPolicies.values().iterator();

while (it.hasNext()) {

javax.management.openmbean.CompositeDataSupport myVal =

(javax.management.openmbean.CompositeDataSupport)it.next();

Iterator it2 = myVal.values().iterator();

while (it2.hasNext()) {

String nextPolicyName = (String)it2.next(); // read name

String nextPolicyDocument = (String)it2.next(); // read policy document

FileUtils.writeFile(export\_directory+"/"+ nextPolicyName+".policy",nextPolicyDocument);

}

}

* ESB\_deployment

import wlstModule

from com.bea.wli.sb.management.configuration import SessionManagementMBean

from com.bea.wli.sb.management.configuration import ALSBConfigurationMBean

from com.bea.wli.config import Ref

from com.bea.wli.config.customization import Customization

from java.io import FileInputStream

from java.util import HashMap

from java.util import ArrayList

from java.util import HashSet

import sys

#=======================================================================================

# Entry function to deploy project configuration and resources

# into a ALSB domain

#=======================================================================================

def importToALSBDomain():

try:

# Declare Variables

sessionMBean = None

alsbConfigurationMBean = None

# Connect to Server

print 'Connecting to server: ', adminUrl

connectToServer(connectMethod)

print 'Starting import of:', importJar, "on ALSB Admin Server:", adminUrl

# Read import jar file

print 'Read import jar file'

theBytes = readBinaryFile(importJar)

print 'Import file read successfully', importJar

# Create unique session name

print 'Creating unique session name'

sessionName = createSessionName()

print 'Created session name :', sessionName

# Create and start session

print 'Creating SessionMBean'

sessionMBean = getSessionMBean(sessionName)

print 'SessionMBean started new session'

# obtain the ALSBConfigurationMBean instance that operates

# on the session that has just been created. Notice that

# the name of the mbean contains the session name.

print 'Create ALSBConfiguration'

alsbConfigurationMBean = findService(String(ALSBConfigurationMBean.NAME

+ ".").concat(sessionName), ALSBConfigurationMBean.TYPE)

print "ALSBConfiguration MBean found", alsbConfigurationMBean

# Perform updates or read operations in the session using alsbSession

# Upload Jar File

print 'Uploading Jar file'

alsbConfigurationMBean.uploadJarFile(theBytes)

print 'Jar Uploaded'

print 'ALSB Project will now get imported'

alsbJarInfo = alsbConfigurationMBean.getImportJarInfo()

alsbImportPlan = alsbJarInfo.getDefaultImportPlan()

alsbImportPlan.setPassphrase(passphrase)

operationMap=HashMap()

operationMap = alsbImportPlan.getOperations()

print 'Default importPlan'

printOpMap(operationMap)

alsbImportPlan.setPreserveExistingEnvValues(false)

alsbImportPlan.setPreserveExistingOperationalValues(false)

print 'Modified importPlan'

printOpMap(operationMap)

importResult = alsbConfigurationMBean.importUploaded(alsbImportPlan)

printDiagMap(importResult.getImportDiagnostics())

if importResult.getFailed().isEmpty() == false:

print 'One or more resources could not be imported properly'

raise

#customize if a customization file is specified

#affects only the created resources

if customFile != None :

print 'Loading customization File', customFile

iStream = FileInputStream(customFile)

customizationList = Customization.fromXML(iStream)

alsbConfigurationMBean.customize(customizationList)

sessionMBean.activateSession(sessionName,

"ALSBImport Operation Completed Successfully")

print "Deployment of : " + importJar + " successful"

except:

print "Unexpected error:", sys.exc\_info()[0]

if sessionMBean != None:

sessionMBean.discardSession(sessionName)

raise

#=======================================================================================

# Utility function to print the list of operations

#=======================================================================================

def printOpMap(map):

set = map.entrySet()

for entry in set:

op = entry.getValue()

print op.getOperation(),

ref = entry.getKey()

print ref

print

#=======================================================================================

# Utility function to print the diagnostics

#=======================================================================================

def printDiagMap(map):

set = map.entrySet()

for entry in set:

diag = entry.getValue().toString()

print diag

print

#=======================================================================================

# Connect to the Admin Server

#=======================================================================================

def connectToServer(connnectMethod):

if connectMethod == "boot":

connect(url=adminUrl, adminServerName=adminServer)

else:

connect(userConfigFile=configFile, userKeyFile=keyFile, url=adminUrl)

domainRuntime()

#=======================================================================================

# Utility function to read a binary file

#=======================================================================================

def readBinaryFile(fileName):

file = open(fileName, 'rb')

bytes = file.read()

return bytes

#=======================================================================================

# Utility function to create an arbitrary session name

#=======================================================================================

def createSessionName():

sessionName = String("ALSBImportScript-"+Long(System.currentTimeMillis()).toString())

return sessionName

#=======================================================================================

# Utility function to load a session MBeans

#=======================================================================================

def getSessionMBean(sessionName):

# obtain session management mbean to create a session.

# This mbean instance can be used more than once to

# create/discard/commit many sessions

sessionMBean = findService(SessionManagementMBean.NAME,SessionManagementMBean.TYPE)

# create a session

sessionMBean.createSession(sessionName)

return sessionMBean

# MAIN section

try:

# import the service bus configuration

importToALSBDomain()

except:

print "Unexpected error: ", sys.exc\_info()[0]

dumpStack()

raise

(www.insemble.com/oracleservicebus-deployment.html)