ORACLE

ORACLE
BUSINESS INTELLIGENCE

Getting started with Weblogic Scripting Tool

From development environment setup to running your WLST scripts

ORACLE WHITE PAPER | OCTOBER 2014

Table of Contents

.

Introduction

"WLST is based on Jython programming language with WebLogic WLST libraries. The execution can happen in the following three modes: scripting, interactive, and embedded."

"OEPE provides tooling for WLST that enable editing, executing, debugging, WebLogic MBean access and navigation, as well as a built-in help for WLST commands."

From Oracle product documentation

This white paper will guide you setting up a development environment, get you started with basic script writing, and point you to useful documentation. We will focus on the interactive and script mode, for embedded mode, please refer to the documentation for examples about instantiating the WLST interpreter in your Java code and using it to run WLST commands and scripts

» Convention in this document:
a sequence of menu and dialog items are separated with 'greater than' sign. Actions included in that sequence will be described between square brackets, e.g.
File > Open... > [select file] > Open

» Assumption: you have a Weblogic server running either locally or on a remote server, know the url and have administrative access to that server.

Up and running in 7 steps

1. download and install OEPE from

http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html

Download the appropriate version for your OS in 'Oracle Enterprise Pack for Eclipse'.

There is an easy way and a slightly more difficult one. The easy way is to use a bundle installer, but not all versions have one. The bundle installer will get you started with everything you need including Weblogic server. The installation is an easy Next>Next>...>Finish.

The other way is slightly more complex, you will need to install Weblogic yourself and some install steps are less easy (step 3, which won't be possible until you create a target runtime, see the eclipse doc here: http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jst.j2ee.doc.user%2Ftopics%2Ftjtargetserver.html).

Tip: unless you have special needs and require a precise version, use a bundle. If there is none for the latest version, you will find bundles on the 'Other versions' page here:
http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/oepe-archive-1716547.html

For other install methods, read the install guide (look for the link on the download page), e.g. http://docs.oracle.com/cd/E50457_02/12131/OEPUG/install.htm

2. create a new project

Start Eclipse, then create a new project.

e.g. *File>New>Project... > JavaEE>Utility Project > Next > [give the project a name] > Finish*

3. add a facet to your project

[right click on the project you created in the project explorer window] > Properties > Project Facets > [check "Oracle WebLogic Scripting Tools (WLST) Support"] > OK

Tip: If you installed without using a bundle, you will need to select a Weblogic server target runtime (or create a new) using the 'Runtime' tab on the right. Only compatible runtimes will be listed, others will not appear or will be grayed out if you check "show all runtimes", so you need to make sure the version you select (drop down) for your WLST support matches the available Weblogic Server version to be able to select a runtime.

You can manage runtimes here: Window > Preferences > Server > Runtime environments

This step will add to the project the WLST interpreter and a directory for your scripts.

4. create a new wlst script

File > New > Other > Oracle > Weblogic > Oracle Weblogic Script > Next > [Select the wlst source folder for your project (created when adding the facet)] > [give it a name, and choose a template]

Alternatively you can *[right click on the wlst directory in your project] > New > Oracle Weblogic Script > [give it a name, and choose a template]*

Templates are optional but useful to get started, and can be selected using the browse button on the right of the template text box in the "New Weblogic WLST script" window. Try them all.

5. change the connection details in the sample script to point to your weblogic server details:

e.g.

```
username = 'me'
password = 'mypassword'
url='t3://myserver.com:7001'
```

6. write your code

Find the WLST reference documentation here:

http://docs.oracle.com > Fusion Middleware > Weblogic server > view Library > Reference > WLST Command Reference for WebLogic Server

e.g. http://docs.oracle.com/middleware/1213/wls/WLSTC/reference.htm

The following book will also help much in understanding WLST

http://docs.oracle.com > Fusion Middleware > Weblogic server > view Library > Books > Understanding the WebLogic Scripting Tool

e.g. http://docs.oracle.com/middleware/1212/wls/WLSTG/index.html

The Jython documentation can be found here.

http://www.jython.org

Remember that Jython is an implementation of Python, most of the Python doc applies.

7. run

(CTRL+F11) or *Run* > *Run* or click the run icon

select *WLST run*

Interactive mode

Interactive mode is an easy way to experiment and try WLST commands.

You can start interactive mode from your operating system as follows:

- Linux
  cd <ORACLE_HOME>/oracle_common/common/bin
  ./wlst.sh
  (replace <ORACLE_HOME> with the path you chose when installing Weblogic. You can also type *locate wlst.sh* to find the command location)
- Windows:
  Start menu > *Oracle WebLogic* > *WebLogic Server* > *Tools* > *WebLogic Scripting Tool*

You can also use the interactive mode from within your OEPE environment:

- select the Console tab, and click on the WLST drop down on the top right of the console window
- select a server, then select the WLST command

Sample interactive mode session:

```
wls:/offline> connect ('me', 'mypassword', 't3://myserver.com:7001')
wls:/bifoundation_domain/serverConfig> ls()
wls:/bifoundation_domain/serverConfig> exit()
```

From the interactive mode, you can record the commands you type to create a basic script:

- Start the interactive mode as explained above
- Type startRecording('<yourFileName.py>')
- Execute WLST commands
- Unless you type the exit() command, stop recording using the stopRecording() command.

You can also record scripts from the Weblogic Administration Console. Detailed information about recording a WLST script from the console can be found in "Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help", e.g.
http://docs.oracle.com/middleware/1213/wls/WLACH/taskhelp/console/RecordWLSTScripts.html

Executing your scripts

Executing a script can be done from the interactive mode, or directly from the command line:

- From the interactive mode, type
  execfile('<yourFileName.py>')
- From the command line, type
  ./wlst.sh <yourFileName.py>
  for linux, or
  wlst.cmd <yourFileName.py>
  for windows.

Include the full script file path if needed.

You can also test your scripts directly from the Eclipse interface:

- *(CTRL+F11)* or *Run* > *Run* or click the run icon > select *WLST run*

Weblogic objects and hierarchical organization

Weblogic mBeans are seen through WLST as a hierarchical organization similar to a file system.

The hierarchy nodes can be browsed through Eclipse interface in the server tab (click on the tab, then on a server name, and start the server if needed using the green triangle on the top right of the server tab).

WLST provide commands to navigate this organization:

Command	Output
ls()	lists current path properties and objects. Read the full reference documentation for the ls() Information function, e.g. http://docs.oracle.com/middleware/1213/wls/WLSTC/reference.htm#WLSTC391
ls('<path>')	lists <path> properties and objects
pwd()	current path
cd('<path>')	move to <path>

Several hierarchies are provided; the following documentation excerpt explains their use:

'If you are using WLST to change the configuration of a security MBean, you must access the edit hierarchy and start an edit session. For example, if you change the value of the LockoutThreshold attribute in UserLockoutManagerMBean, you must be in the edit hierarchy.

If you invoke security provider operations to add, modify, or remove data in a security provider data store, WLST does not allow you to be in the edit hierarchy. Instead, invoke these commands from the serverConfig or domainConfig hierarchy. For example, you cannot invoke the createUser operation in an AuthenticatorMBean MBean from the edit hierarchy. WLST enforces this restriction to prevent the possibility of incompatible changes. For example, an edit session could contain an unactivated change that removes a security feature and will invalidate modifications to the provider's data.'

'WLST first connects to a WebLogic Server instance at the root of the server's configuration MBeans, a single hierarchy whose root is DomainMBean.'

The table below provides some WLST commands needed to change hierarchy root:

Command	Effect
serverRuntime()	Location changed to serverRuntime tree. This is a read-only tree with ServerRuntimeMBean as the root.
domainRuntime()	Location changed to domainRuntime tree. This is a read-only tree with DomainRuntimeMBean as the root.
serverConfig()	Location changed to serverConfig tree. This is a read-only tree with DomainMBean as the root.
Edit()	Location changed to edit tree. This is a writable tree with DomainMBean as the root. To make changes you will need to start an edit session via startEdit().
domainCustom()	Navigates to the root of domain custom MBeans that are registered in the Domain Runtime MBeanServer. WLST navigates, interrogates, and edits domain custom MBeans; however, domain custom MBeans cannot use the cmo variable because a stub is not available. The domainCustom command is available when WLST is connected to an Administration Server instance. Sample use is documented in http://docs.oracle.com > Fusion Middleware > Weblogic server > view Library > Books > Development > Developing Custom Management Utilities Using JMX for Oracle WebLogic Server e.g. http://docs.oracle.com/middleware/1213/wls/JMXCU/understandwls.htm#JMXCU239

cmo variable

The cmo (Current Management Object) variable is provided by WLST, and points to the current mBean you navigated to. You can use the set and get methods to interact with the mBean properties, see sample code below.

Other useful variables are provided. All variables created by WLST are listed in the reference documentation for WLST, e.g. http://docs.oracle.com/middleware/1213/wls/WLSTC/reference.htm#i1003217

Sample commands:

```
# connect to the server (online mode)
connect (‘me’, 'mypassword', 't3://myserver.com:7001')
# display current root
pwd()
# move to the SQLGroupProvider mBean
cd ('SecurityConfiguration/bifoundation_domain/DefaultRealm/myrealm/AuthenticationProviders/SQLGroupProvider')
# get the DataSourceJNDIName property
cmo.getDataSourceJNDIName()
# move to the edit tree
Edit()
#set the AdministrationPort property value to 9092
cmo.setAdministrationPort(9092)
```

Online or Offline modes

The paragraph titled “Using WLST Online or Offline’ in ‘Understanding the WebLogic Scripting Tool’ book (http://docs.oracle.com/middleware/1212/wls/WLSTG/using_wlst.htm#WLSTG119) discusses when to use the online or offline mode in WLST. This paper mainly explains the Online mode use.

When starting WLST, Offline is the default mode. Online is started when using the connect() command successfully, e.g. *connect (‘me’, 'mypassword', 't3://myserver.com:7001')*

Writing in Jython, code samples

Jython is an implementation of Python. If you are not familiar with Python, you will find plenty of code examples on the web. https://www.python.org/ is a good place to start

Python functions have no explicit begin or end, and no curly braces to mark where the function code starts and stops. The only delimiter is a colon (:) and the indentation of the code itself. Be very careful with your code indentation.

Use # before comments. Within the Eclipse IDE, the <Ctrl><Shift>/ will comment out the selected text, or un-comment it.

Here is a selection of useful commands to get started:

print

print <anything>

```
myVar = 'world'
print 'Hello', myVar
```

will print *Hello world* to the default output (console).

```
severConfig()
print cmo.getDomainVersion()
```

will print the DomainVersion property value to the defaut output.

for loop

Python uses interesting data types through which you can browse using for loops, e.g.:

```
for color in ['red', 'blue', 'green']:
        print color

fo = open("myFile ", "r")
for line in fo.readlines():
        print line
```

if then statement

This is a frequently used statement in any code. Remember the semicolons and be careful with the indentation.

```
if weather == 'fair':
        print 'go for a walk"
else:
        print 'stay home'
print 'this will always print, whatever the weather value'
```

Exception handling

Jython provides exception handling techniques that will allow your scripts to handle errors gracefully. The Jython documentation about exception handling is here:
http://www.jython.org/jythonbook/en/1.0/ExceptionHandlingDebug.html

This is a basic code exemple:

```
try:
        connect('me','myPasword','t3://myServer.com:7001')
except:
        print '\tUnable to connect to admin server...'
        exit()
```

Recursive programming

The following function is recursively listing the MBeans hierarchy and properties. It is initially called from main.

```
def ListBranchNodesAndProperties(path):
```

```
    print "\n",path
    ls(path,returnMap = 'true',returnType='a') # lists node properties
    subNodes = ls(path,returnMap='true',returnType='c') # lists all child nodes
    if subNodes: # empty collections have a boolean value of 'false'
        for n in subNodes:
            if n not in path: # trees can be recursive, so stop if node has already been found in branch
                ListBranchNodesAndProperties(path+n+'/')
    return

if __name__== "main":
    print 'starting the script ....'
    username = 'me'
    password = 'mypassword'
    url='t3://localhost:7001'
    connect(username,password,url)
    domainConfig()
    cd('Log')
    print pwd()
    ListBranchNodesAndProperties('')
    print 'Done!'
```

Getting and Setting MBean attributes

This can be done using either the *set('attrName',value)* / g*et('attrName',value)* syntax, or *cmo.setattrName(value) / cmo.getattrName(value)*

e.g.: *set('ListenPort', 7011)* is equivalent to *cmo.setListenPort(7011)*

```
    print 'starting the script ....'
    username = 'me'
    password = 'mypassword'
    url='t3://localhost:7001'

    connect(username,password,url)

    edit()
    cd('Servers')
    ServerBean = lookup('myserver')
    oldValue = ServerBean.getStuckThreadMaxTime()
    newValue = 800
    print 'StuckThreadMaxTime',oldValue
    startEdit()
```

```
# set is only possible in the edit tree, must be preceded with startEdit, and followed with activate()
    ServerBean.setStuckThreadMaxTime(newValue)
    activate()
    print 'StuckThreadMaxTime has been changed from', oldValue, 'to', newValue
    print 'Please restart the server'
```

Using libraries

Importing external libraries (often called modules in python) in full or in part is straightforward. Jython also allows easy reuse of Java libraries.

This Jython book chapter covers the technique in detail:

http://www.jython.org/jythonbook/en/1.0/ModulesPackages.html

Also do not miss to read this other chapter about Jython and Java integration:
http://www.jython.org/jythonbook/en/1.0/JythonAndJavaIntegration.html

Below are a few simple code examples:

» Importing the OS module which provides a portable way of using operating system dependent functionality:

```
Import os
```

» Importing only some objects from a java library:

```
from java.io import FileInputStream # used for reading properties file
def main()
        try:
                propertiesInputStream = FileInputStream("sample.properties")
        except:
                print '\tCan"t find sample.properties file. Make sure the file is located in the script directory.'
         exit()
```

» Another partial import example to use jdbc

```
from com.ziclix.python.sql import zxJDBC
try:
        db = zxJDBC.connect('jdbc:oracle:thin:@server.oracle.com:1521:orcl', 'user', 'password', 'oracle.jdbc.OracleDriver')
except:
        print '\tCould not connect to the database for the following reason:'
        print sys.exc_info()[1]
else:
        c = db.cursor()
…
```

Going further with Jython / Python

Style Guide

If the python code you write is for distribution (most often the case than not), you will want to make sure your syntax is clean and readable by others, and follows Python style guide, aka PEP8 http://legacy.python.org/dev/peps/pep-0008/.

OEPE actually includes PyDev (http://pydev.org/), which among many other useful features will help you by checking your code against PEP8 rules, and can also auto correct some of it:

» Activating PEP8 checking in Eclipse: Window > Preferences > PyDev > Editor > Code Analysis > PEP8.py > Warning

Warning signs will appear in Eclipse on the left of your code lines with the precise description of the style infringement. Eventually, use [right click anywhere in your code] > PyDev > Code Analysis.

Some of the cleanup can be done for you by PyDev

» Window > Preferences > PyDev > Editor > Code Style > Code Formatter > [check 'Auto-format editor contents before saving?']

This will tidy up some of it each time you save your code. If you want to go further and have all the style fixes done for you, check out this link: http://stackoverflow.com/questions/11046001/is-there-any-way-to-fix-pep-8-issues-with-pydev

Importing WLST as a Jython module

This will allow using WLST directly from Jython. All WLST commands will be available from your Jython program. See http://docs.oracle.com/middleware/1212/wls/WLSTG/using_wlst.htm#i1093409.

Conclusion

WLST is a powerful tool, useful for automating Weblogic administrative tasks, running diagnostics scripts. Integrated with the Python language, it will allow virtually anything.

Trying to embrace the full capabilities of this association can be daunting, and mastering the finest Python intricacies may seem overwhelming, but the language allows a very progressive learning curve, and once the hierarchical structure exposed by WLST is understood and some basic Python programming mastered, you will be set for quick wins. You are not alone, and the web is full of code snippets, tutorials, and other documentation to support you during that journey.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together