

# Building WebLogic Domains with WLST

Matt Brasier – C2B2 Consulting LTD



# Matt Brasier

- Consultant at C2B2
  - Middleware support and services
- 13 years experience with WebLogic
- Specialise in high performance Java applications
- Author



## Oracle SOA Suite 11g Performance Tuning Cookbook

Discover how you can get the best performance from your  
Oracle SOA Suite 11g infrastructure.

Matthew Brasier Nicholas Wright  enterprise  
Publishing



# Agenda

- Problem Statement
- What is WLST?
- Why should we automate domain builds?
- How should we do it?

# Problem Statement

- Increasing demand for repeatable builds
  - Don't just consider the application build
- Repeatable environment builds
  - Scaling out
  - New Environments
  - New projects
  - Disaster Recovery

# What is WLST?

- **WebLogic Scripting Toolkit**
  - Introduced in WebLogic 9.0
- **Jython based scripting language for interacting with WebLogic**
  - Anything you can do through the WebLogic console
  - And more...
- **Better than other vendor equivalents**
  - Full power of python based scripting
    - Logic, File IO, network IO etc

# How to use WLST

- Distributed with WebLogic
  - `$FMW_HOME/wlserverXX/common/bin/wlst.sh`
  - `%FMW_HOME%/wlserverXX/common/bin/wlst.cmd`
- Execute on its own to launch in interactive mode
  - Very useful for debugging and discovering resources you need
- Specify a file name to execute a script

# What can we do with WLST?

- View/Edit configuration
  - Server properties
  - Connection pools
  - JMS resources
  - Applications
  - Clustering

# What can we do with WLST?

- Read runtime metrics
  - Message queue sizes
  - Memory use
  - GC stats
  - Call stats
  - Anything available via JMX



# What can we do with WLST?

- Operations
  - Start servers
  - Stop servers
  - Deploy applications
  - Reset connection pools
  - Clear JMS queues

# WLST Basics – by example

# Connet to the admin server

```
connect('weblogic','password1','t3://localhost:7001');
```

#Lock configuration and start editing

```
edit();
```

```
startEdit();
```

# Navigate to the server start properties for FORMS\_SERVER

```
cd('Servers/FORMS_SERVER/ServerStart/FORMS_SERVER');
```

# WLST Basics – by example

## #Set the properties

```
set('Arguments', '-Xms=2048m -Xmx=2048m -XX:NewSize=512M -XX:MaxNewSize=512M -  
    XX:PermSize=256M -XX:MaxPermSize=256M');  
set('Username', 'weblogic');  
set('Password', 'welcome1');  
set('BeaHome', os.getenv["WL_HOME"]); //This needs an import that we have not shown  
set('JavaHome', os.getenv["JAVA_HOME"]); //This needs an import that we have not shown  
set('JavaVendor', 'BEA');
```

## #Save changes and apply

```
save();  
activate();
```

# WLST Things to watch out for

- Python
  - Whitespace/indentation matters
  - String types (‘ ‘ vs “ “)
- Security
  - You can use a secure key/password store rather than username/password

# WLST Things to watch out for

- MBean trees
  - WLST splits the configuration into different “trees”
  - Switch to the appropriate tree
    - ServerRuntime, DomainConfig, ServerConfig, Custom

# Example WLST script

```
import sys
import datetime
def printf(format, *args):
    sys.stdout.write(format % args)
connect('weblogic','welcome1','t3://localhost:7001')
servers = domainRuntimeService.getServerRuntimes();
if (len(servers) > 0):
    print 'Time, Server, Destination, ConsumersCurrentCount, MessagesCurrentCount, MessagesHighCount,
    MessagesPendingCount, MessagesReceivedCount';
    while True:
        for server in servers:
            jmsRuntime = server.getJMSRuntime();
            jmsServers = jmsRuntime.getJMSServers();
            for jmsServer in jmsServers:
                destinations = jmsServer.getDestinations();
                for destination in destinations:
                    printf('%s, %s, %s, %i, %i, %i, %i, %i\n',datetime.datetime.now().time(), server.getName(),
                    destination.getName(), destination.getConsumersCurrentCount(), destination.getMessagesCurrentCount(),
                    destination.getMessagesHighCount(), destination.getMessagesPendingCount(), destination.getMessagesReceivedCount());
            java.lang.Thread.sleep(1000);
```

# Recording WLST scripts

- You can record WLST scripts using the WebLogic admin console
  - Any changes you make get written to a script
  - Hard codes in variables (URLs etc)
  - Can be a good basis for change scripts

# Recording WLST

The screenshot displays the Oracle WebLogic Server Administration Console interface. The browser window title is "Home Page - soa\_perf\_domain - WLS Console - Mozilla Firefox". The address bar shows the URL: `localhost:7001/console/console.portal?_nfpb=true&_pageLabel=http://localhost:7001/console/console.portal?_nfpb=true&_pageLabel=HomePage1`. The page header includes the Oracle logo, "WebLogic Server Administration Console", and navigation links for Home, Log Out, Preferences, Record, and Help. A "Start Recording" button is highlighted in the top navigation area. The main content area is titled "Home Page" and contains several sections: "Information and Resources" with helpful tools and general information; "Domain Configurations" with a tree view for Domain, Environment, and Services; "Your Deployed Resources" showing Deployments; and "Your Application's Security Settings" showing Security Realms. On the left side, there are panels for "Change Center" (with Lock & Edit and Release Configuration buttons), "Domain Structure" (showing a tree view of soa\_perf\_domain), "How do I...?" (with search and help links), and "System Status" (showing Health of Running Servers with a bar chart indicating 1 OK status).



# Recording WLST

The screenshot displays the Oracle WebLogic Server Administration Console in a Mozilla Firefox browser window. The page title is "Settings for soa\_server1 - soa\_perf\_domain - WLS Console". The browser address bar shows a localhost URL. The console interface includes a navigation menu on the left with sections like "Change Center", "Domain Structure", "How do I...", and "System Status". The main content area is titled "Settings for soa\_server1" and contains a "Configuration" tab with sub-tabs for General, Cluster, Services, Keystores, SSL, Federation Services, Deployment, Migration, Tuning, Overload, Health Monitoring, Server Start, and Web Services. The "General" sub-tab is active, showing configuration fields for Name (soa\_server1), Machine (LocalMachine), Cluster (Stand-Alone), Listen Address, Listen Port (8101), SSL Listen Port (8002), Client Cert Proxy Enabled, Java Compiler (javac), and Diagnostic Volume (Low). A message at the top states "The recording session has ended. Script recorded to E:\Oracle\Middleware\user\_projects\domains\soa\_perf\_domain\Script1415730644792.py." The footer contains copyright information for Oracle and C2B2 Consulting Limited 2013.

# Recorded WLST

```
startEdit()
```

```
cd('/Servers/soa_server1')
```

```
cmo.setListenPort(8101)
```

```
cmo.setCluster(None)
```

```
activate()
```

# Automating domain building

- Why automate a domain build?
  - Consistency between environments
  - Build new environments quickly
  - Disaster Recovery
  - Consistency over time

# Building a domain

- Key steps
  - Install software
  - Create domain
  - Customise configuration

# Install software

- Not done with WLST
- Automate using silent installers for the relevant products
  - No standard approach across oracle products
  - Response files used to answer the installer “Questions”
    - Can generate the response files

# Create domain

- A number of options available
  - WLST script creates from a domain template
  - Configuration wizard creates from a domain template
  - Create with a WLST script generated from an existing domain
  - Write a WLST script from scratch

# Create base domain

- Create from template
  - WLST *createDomain* command
  - Template can be re-used by configuration wizard
    - And pack/unpack commands
  - Can open template 'offline' and edit it
  - Not human readable
  - Hard to edit the template
  - Brittle to version changes

# Create base domain

- Create a script from existing domain
  - *WLST configToScript* command
  - Script can only be run by WLST
  - Script can be edited to customise it



# configToScript

```
wls:/offline> configToScript("E:\Oracle\Middleware\user_projects\domains\soa_perf_domain");
configToScript is loading configuration from E:\Oracle\Middleware\user_projects\domains\soa_perf_domain\config\config.xml ...
Completed configuration load, now converting resources to wlst script...
Creating the key file can reduce the security of your system if it is not kept in a secured location after it is created. Creating
new key...
Using existing user key file...
Using existing user key file...
.
Using existing user key file...
Using existing user key file...
Using existing user key file...
Using existing user key file...
Using existing user key file...
configToScript completed successfully The WLST script is written to E:\Oracle\Middleware\user_projects\domains\soa_perf_domain\con
fig\config.py and the properties file associated with this script is written to
    E:\Oracle\Middleware\user_projects\domains\soa_perf_domain\config\config.py.properties
WLST found encrypted passwords in the domain configuration.
These passwords are stored encrypted in E:/Oracle/Middleware/user_projects/domains/soa_perf_domain/config/c2sConfigsoa_perf_domain

and E:/Oracle/Middleware/user_projects/domains/soa_perf_domain/config/c2sSecretsoa_perf_domain. WLST will use these password values
while the script is run.
wls:/offline>
```

# configToScript

```
def create_WLDFSystemResource_14(path, beanName):
  cd(path)
  try:
    print "creating mbean of type WLDFSystemResource ..."
    theBean = cmo.lookupWLDFSystemResource(beanName)
    if theBean == None:
      cmo.createWLDFSystemResource(beanName)
  except java.lang.UnsupportedOperationException, usoe:
    pass
  except weblogic.descriptor.BeanAlreadyExistsException, bae:
    pass
  except java.lang.reflect.UndeclaredThrowableException, udt:
    pass
```

# Customise configurations

- Once the domain is created you can customise it further with WLST
- Standard WLST commands
  - Set attributes
  - Create MBeans
  - Set Options
  - Deploy applications

# Reading MBean values

- Reading existing values in scripts allows you to make changes based on existing values
  - 20% increase
  - mycompany.com -> dev.mycompany.com

# Advanced customisation

- Build a domain from the ground up
  - Start with just the admin server
  - Add all other resources using WLST
  - Create a framework that reads a set of scripts
    - New changes can be done with scripts and dropped in to the framework for new deployments
    - All scripts read from a single properties file

# Summary

- WLST provides a number of ways to build and configure your WebLogic domain
  - It won't install the software though
- Changes to domain configuration can be applied with WLST scripts rather than through the console
  - Can be repeated across environments

# Questions?



- @mbrasier, @c2b2consulting