

# Oracle DBA & Developer Days 2011

日本オラクル、今年最大の技術トレーニングイベント

2011年11月9日(水)～11月11日(金) シェラトン都ホテル東京



## ORACLE®

### サポートエンジニアが語る！ WebLogic Scripting Toolを 活用したWLSの管理・監視方法

日本オラクル株式会社 カスタマーサポートサービス統括 テクノロジーサポート本部  
シニアテクニカルサポートエンジニア 小西紀行

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント(確約)するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

# アジェンダ

- WebLogic Scripting Tool(WLST) とは？
  - WLST とはどのようなものか
  - WLST はどのように動いているか
- WLST でどのようなことができるか
  - 管理コンソールと同じことが WLST でもできる
  - スクリプトによる自動化
- 必要な情報の探し方、WLST でのアクセス方法
  - 管理コンソールと同じことをどう WLST で行うか
  - 必要な情報を探して WLST からアクセスする流れ
- 事例紹介、デモ
  - WLST を使用した WLS の監視、通知
  - WLST を使用した WLS の制御(シャットダウン)

# WebLogic Scripting Tool(WLST) とは？

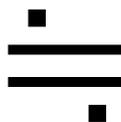
- WebLogic Scripting Tool(WLST) とは？
  - WebLogic Server(WLS) 管理用のコマンドライン、スクリプト環境
  - WLS が提供する MBean へアクセス
    - MBean の構造や種類、ツリーによる管理構造
    - 管理コンソールと同様の操作
  - 管理コンソールで行っている作業の効率化、自動化
- WLST でどのようなことができるか
- 必要な情報の探し方、WLST でのアクセス方法
- 事例紹介
- デモ

# WLST とは？

管理コンソールと同様に、WLS の管理、操作に使用できる  
コマンドライン スクリプト環境



管理コンソール画面



```
C:¥>java weblogic.WLST
```

```
Initializing WebLogic  
Scripting Tool (WLST) ...
```

```
wls:/offline>connect ()
```

WLST 実行画面

管理コンソールでできることは、WLST でもできる！

# WLST とは？

コンソール内で画面遷移を繰り返して毎回行っていた操作

- 目的の情報に直接移動、より早いレスポンス
- WLST の提供するコマンド・変数による操作

サーバ1 → 構成  
→ 全般 → サー  
バ2 → 構成  
→ … サーバ3  
→ …



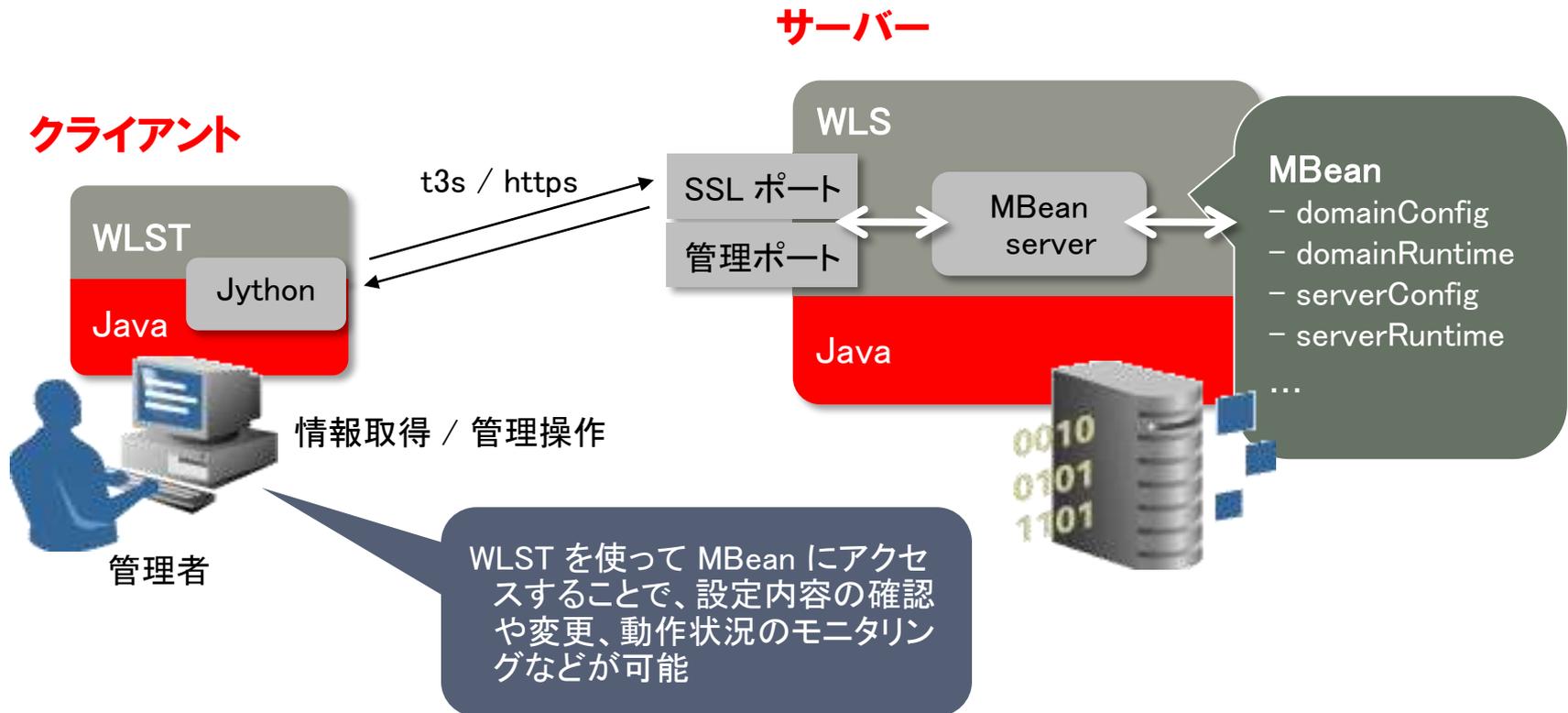
```
WLST: > cd('domain/server/runtime')
WLST: > edit('Script131903952600')
ロケーションが編集ツリーに移動しました。これは、DomainMBeanをルートとする
読み込み可能なツリーです。変更するには、startEdit()を使用して
編集セッションを開始する必要があります。
詳細は、help(ed())を使用してください
編集セッションを開始しています ...
編集セッションが開始されました。変更が完了したら、必ず保存してアクティブ化
してください。
すべての変更をアクティブ化しています。しばらく時間がかかる場合があります ...
アクティブ化が完了すると、この編集セッションに開通付けられた
編集ロックが開放されます。
アクティブ化が完了しました
WLST: > cd('domain/server/runtime')
WLST: > cd('Data_Sources')
```

スクリプトに作業内容を  
まとめることで

WebLogic Server の管理作業を自動化できる！

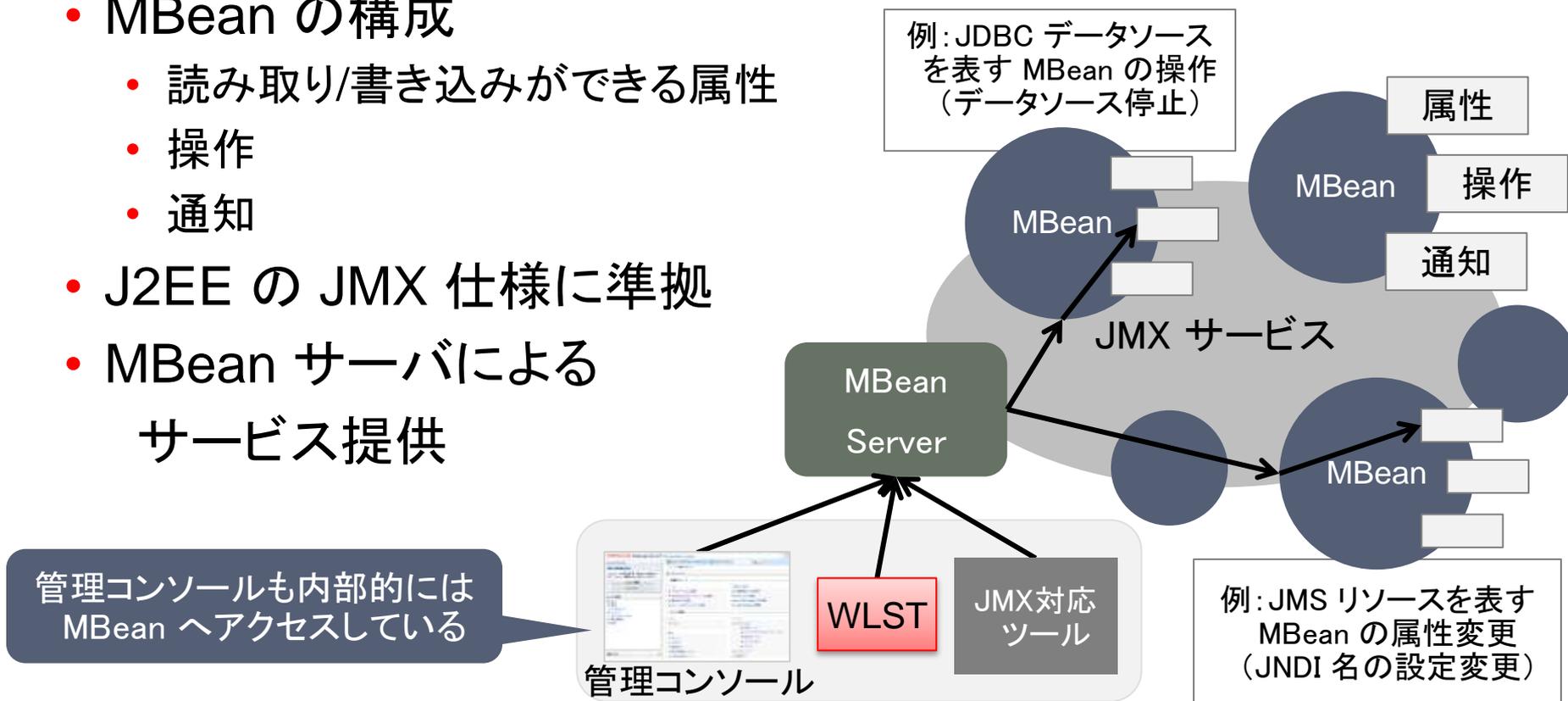
# WLST とは？

- Jython(Pure Java の Python 実装) ベース
- WLS の MBean へアクセスして、情報取得、管理操作が可能



# WLST とは？ - MBean とは？

- MBean = 管理対象 Bean(Managed Bean)
  - 管理されるリソースを表す Java オブジェクト
- MBean の構成
  - 読み取り/書き込みができる属性
  - 操作
  - 通知
- J2EE の JMX 仕様に準拠
- MBean サーバによるサービス提供



# WLST とは？ - MBean とは？

## [GUI (Jconsole) の画面]

The screenshot shows the Java Monitoring & Management Console (JConsole) interface. On the left, a tree view displays the MBean hierarchy under 'pid: 4488 weblogic.Serv'. The 'ServerRuntime' MBean is selected, and its sub-items '属性' (Attributes), '操作' (Operations), and '通知' (Notifications) are visible. A callout bubble points to this tree with the text: 'OS のディレクトリ構造のように、ツリー構造で管理' (Managed in a tree structure like the OS directory structure). Below the tree, a table lists the attributes of the selected MBean. A callout bubble points to the table with the text: '各MBean の持つ属性値、サーバの設定情報' (Attribute values of each MBean, server configuration information). Another callout bubble points to the '属性' sub-item in the tree with the text: 'WLS リソースを表す MBean' (MBean representing WLS resources).

OS のディレクトリ構造のように、ツリー構造で管理

名前	値
AdministrationPortEnabled	false
AdministrationURL	t3://10.185.177.206:7001
ApplicationRuntimes	javax.management.ObjectName[12]
AsyncReplicationRuntime	
ClusterMaster	false
ClusterRuntime	
ConnectorServiceRuntime	com.bea:ServerRuntime=AdminServer,Name=...
CurrentDirectory	C:\bea2\10.3.5\user_projects\domains\JS...
CurrentMachine	
DefaultExecuteQueueRuntime	
DefaultURL	t3://10.185.177.206:7001
EntityCacheCumulativeRuntime	
EntityCacheCurrentStateRuntime	
EntityCacheHistoricalRuntime	

WLS リソースを表す MBean

各MBean の持つ属性値、サーバの設定情報

# WLST とは？ - MBean とは？

ツリー構造が UNIX のディレクトリ構造と同じように表現

[WLST 画面]

```
C:\ コマンド プロンプト - java weblogic.WLST
wls:/PlanTest/serverConfig> serverRuntime()
wls:/PlanTest/serverRuntime> pwd()
'serverRuntime:/'
wls:/PlanTest/serverRuntime> cd('ApplicationRuntimes')
wls:/PlanTest/serverRuntime/ApplicationRuntimes> pwd()
'serverRuntime:/ApplicationRuntimes'
wls:/PlanTest/serverRuntime/ApplicationRuntimes> ls()
dr--  bea_wls9_async_response
dr--  bea_wls_deployment_internal
dr--  bea_wls_diagnostics
dr--  bea_wls_internal
dr--  bea_wls_management_internal2
dr--  consoleapp
```

pwd() : 現在の階層を確認

cd() : 指定した階層に移動

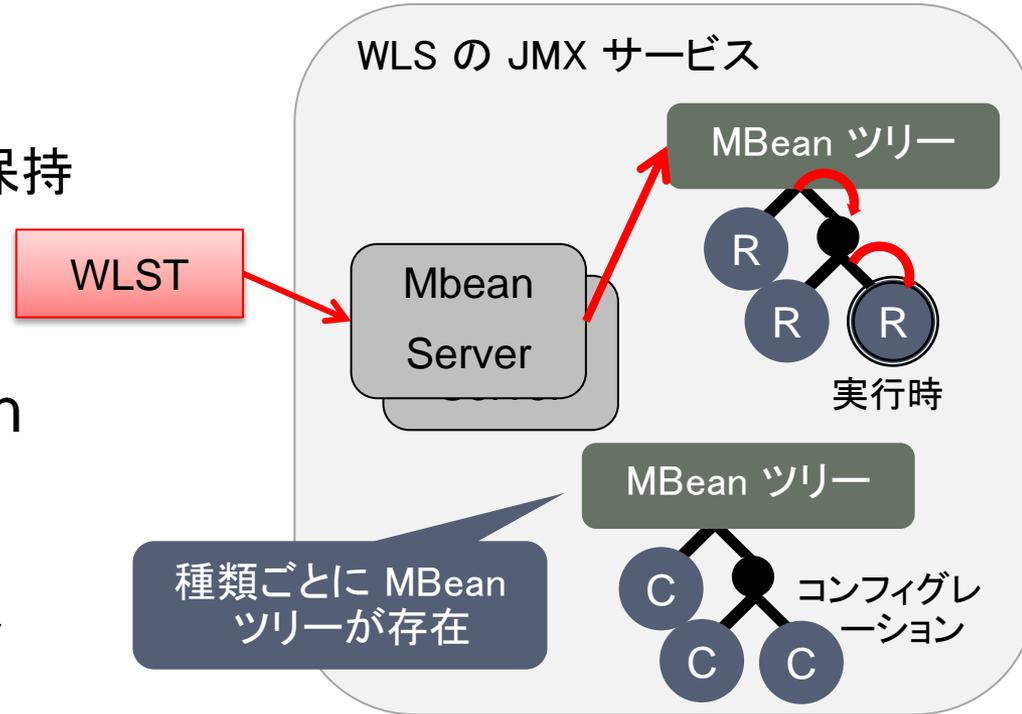
ls() : MBeanの情報を参照

UNIX と同様のコマンドで移動できる

# WLST とは？ - MBean とは？

## WLS の MBean の種類

- 実行時 MBean
  - 現在の状況に関する情報を保持
    - サーバの状態
    - 現在のスレッド数 など
- コンフィグレーション MBean
  - 設定に関する情報を保持
    - データソースの最大容量
    - サーバの起動モード など



WLST では、実行時 MBean とコンフィグレーション MBean で分かれてツリー構造で管理されている

# WLST とは？ - MBean とは？

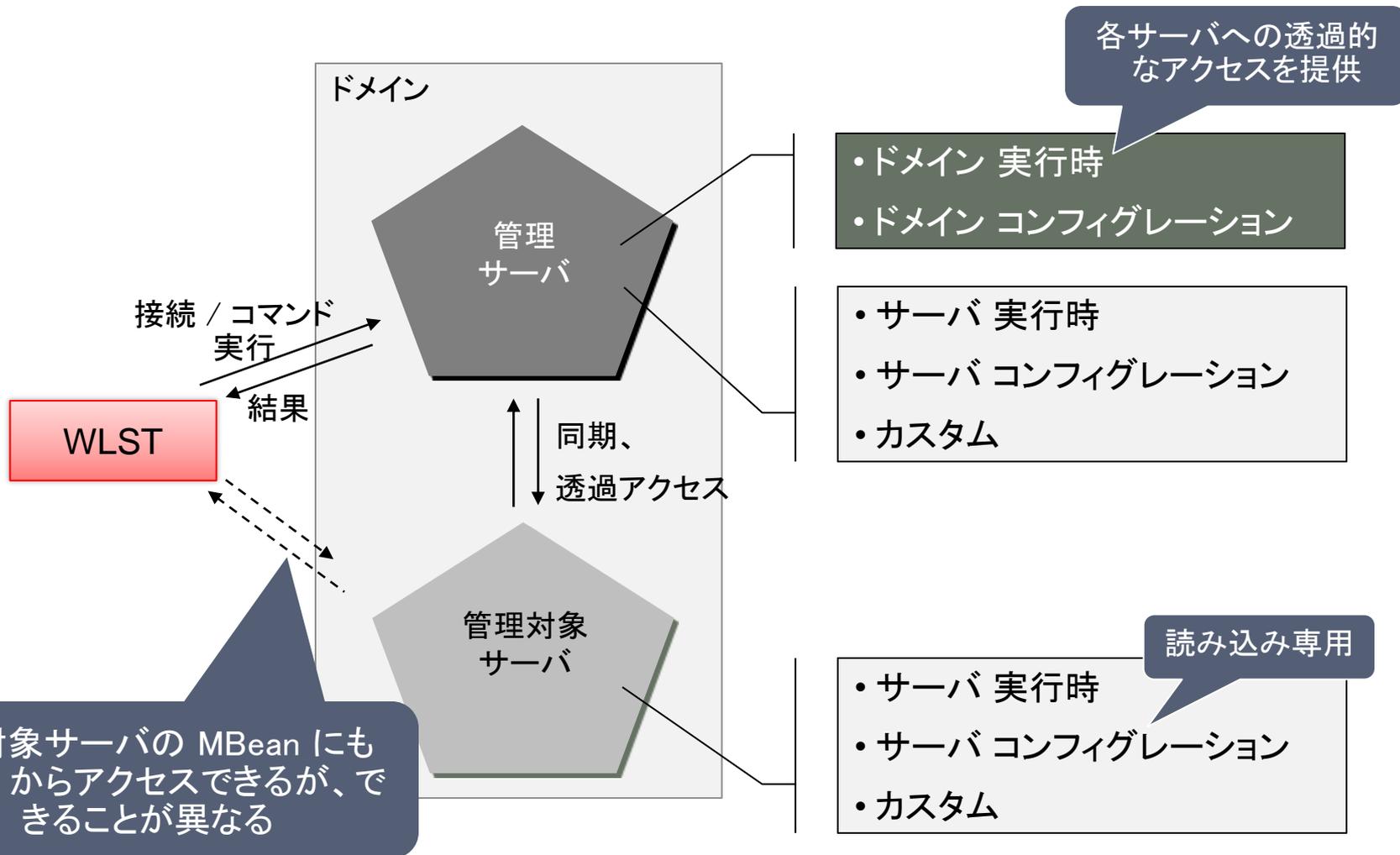
## WLS の MBean ツリーの種類

- ドメイン実行時 MBean ツリー
  - ドメインコンフィグレーション MBean ツリー
  - サーバ実行時 MBean ツリー
  - サーバコンフィグレーション MBean ツリー
  - カスタム MBean ツリー
- ドメイン全体の情報
  - 管理サーバにのみ存在
  - 自サーバの情報
  - すべてのサーバに存在
  - Java VM の MBean やアプリケーションが登録する MBean
  - すべてのサーバに存在

WLST コマンドでツリーを切り替えて目的の MBean にアクセス

```
domainRuntime(),  
domainConfig(),  
serverRuntime() ...
```

# WLST とは？ - MBean とは？

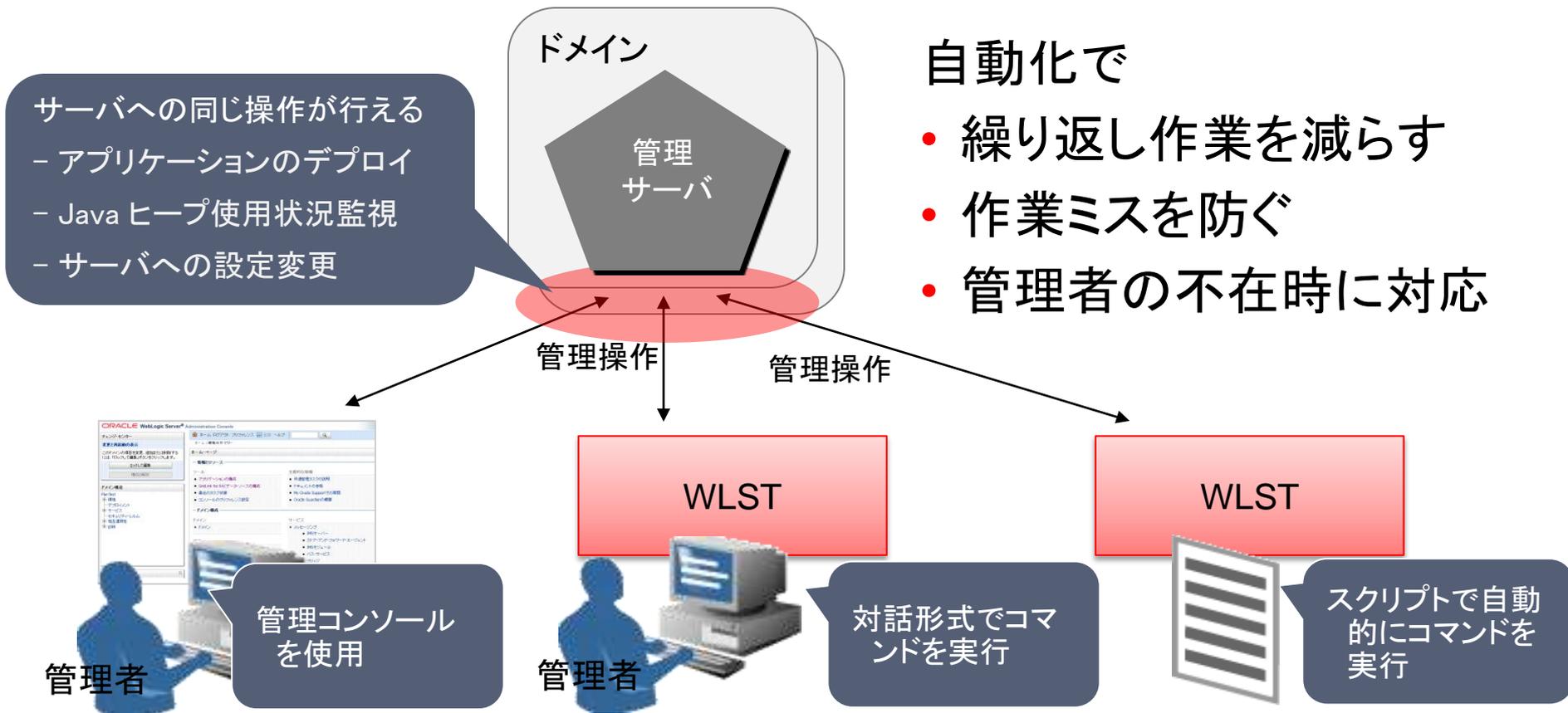


# WLST でどのようなことができるか

- WebLogic Scripting Tool(WLST) とは？
- WLST でどのようなことができるか
  - 管理コンソールと同じことが WLST でもできる
  - スクリプトによる自動化
    - WLST コマンドや変数、ライブラリの利用
  - 作業に応じて WLST のモードを選択して実行
    - 対話実行モード/スクリプト実行モード
    - オフラインモード/オンラインモード
- 必要な情報の探し方、WLST でのアクセス方法
- 事例紹介
- デモ

# WLST を使ってどのようなことができるか？

- 管理コンソールでできることは、WLST でもできる
- WLST のコマンドライン環境で、手順の自動化、定期実行



# WLST を使ってどのようなことができるか？

## WLST 実行モード

- 対話形式での実行
  - `java weblogic.WLST`
- スクリプト形式での実行
  - `java weblogic.WLST XXXX.py`
  - スクリプトを実行後、対話形式に戻る
    - `java weblogic.WLST -i XXX.py`



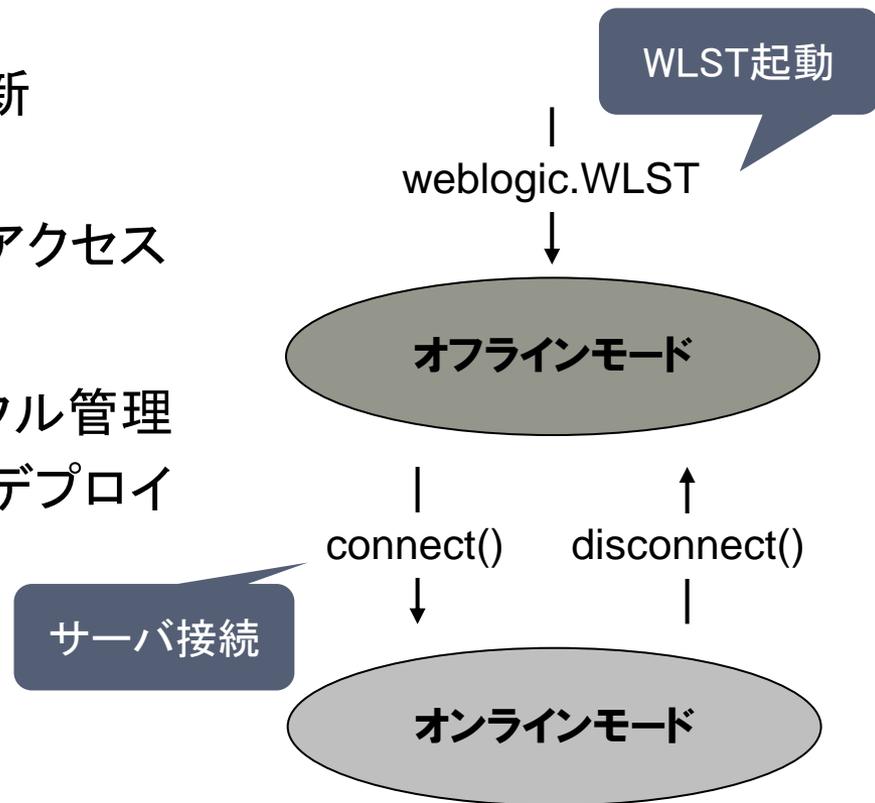
# WLST オフライン/オンラインモード

- オフラインモード

- サーバインスタンスへの接続
  - ドメインの作成/既存ドメインの更新
  - ドメインテンプレートの作成 など
- コンフィグレーション Bean へのアクセス

- オンラインモード

- サーバインスタンスのライフサイクル管理
  - アプリケーションのデプロイ・アンデプロイ
  - 設定情報の参照・更新 など
- 実行時 MBean、コンフィグレーション MBean へのアクセス



# スクリプトの利用による自動化

- WLST の提供するコマンド
  - connect(), startEdit(), deploy() ...
- Java の提供する機能が使用可能

オンラインマニュアル  
『WLST コマンドおよび  
変数リファレンス』

```
・パッケージのインポート  
  from java.lang import *  
  import javax.swing as swing  
  
・オブジェクトの作成  
  str = String("Test String")  
  
・メソッドの呼び出し  
  print System.currentTimeMillis()
```

- Python の提供する機能が使用可能
  - 文字列操作/ファイル操作/ユーザ入力/ループ、比較 → 補足資料を参照

# スクリプトの利用による自動化

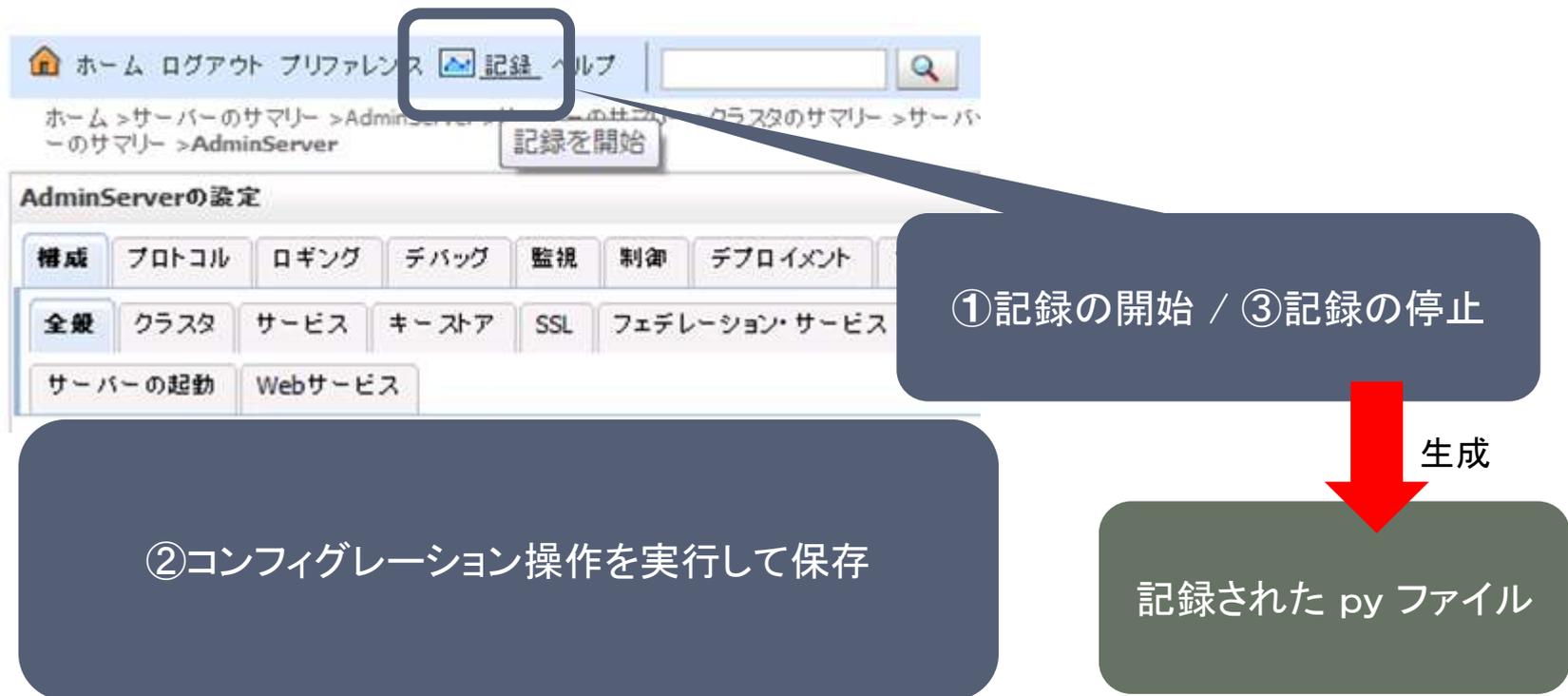
- cmo =現在の MBean オブジェクトを表す変数
  - (current management object)
- `serverRuntime()` → `cmo = ServerRuntimeMBean`
- MBean の持つAPIを呼び出し可能
  - `cmo.shutdown()`
  - `cmo.getState()`
- 他の変数に代入して利用可能
  - `sruntime = cmo`
  - `sruntime.getState()`

サーバ実行時 Mbean ツリーに移動すると、現在のcmoには ServerRuntimeMBeanが自動的に入っている

呼び出しには“cmo”が必要

# 管理コンソールでの操作の記録

- 管理コンソール上の実際の設定変更作業を記録してスクリプトとして編集、再利用が可能



# 必要な情報の探し方、WLST でのアクセス方法

- WebLogic Scripting Tool(WLST) とは？
- WLST でどのようなことができるか
- **必要な情報の探し方、WLST でのアクセス方法**
  - 管理コンソールからの MBean、属性確認
  - リファレンスマニュアルでの詳細な機能の調査
  - MBean がどのツリーで管理されているかを把握
  - WLST で MBean を検索して、ツリーを移動
- 事例紹介、デモ

# WLST でどのように実現するか

[WLST を使用する管理者のよくある疑問]

- ・ どのように目的のMBeanや属性を見つけたらよいか。
- ・ 管理コンソールと同じ情報ことをWLSTでどうやるか。
- ・ 見つけたMBeanをどのように呼び出したらよいか。



```
コマンドプロンプト - java weblogic.WLST
wls:/PlanTest/serverConfig> serverRuntime()

wls:/PlanTest/serverRuntime> pwd()
'serverRuntime:/'
wls:/PlanTest/serverRuntime> cd('ApplicationRuntime')
wls:/PlanTest/serverRuntime/ApplicationRuntime> pwd()
'serverRuntime:ApplicationRuntime'
wls:/PlanTest/serverRuntime/ApplicationRuntime> ls()
dr-- bea_wls_async_response
dr-- bea_wls_deployment_internal
dr-- bea_wls_diagnostics
dr-- bea_wls_internal
dr-- bea_wls_management_internal?
```

WLST では、MBean を見て作業を行う。種類によって管理が分かれており、あらかじめ把握が必要

# 目的の MBean をどう見つけるか

## 管理コンソールのヘルプ画面が有効

1. 管理コンソール
2. オンラインマニュアル

オンラインマニュアルからも  
参照可能

The screenshot shows the Oracle JMX console interface. A red arrow labeled 'Click' points to the 'ヘルプ' (Help) button in the top navigation bar. The main content area is titled 'ヘルプ画面' (Help Page) and 'サーバー: 監視: パフォーマンス' (Server: Monitor: Performance). It contains a table with the following data:

名前	説明
現在の空きヒープ	JVMヒープで現在使用可能なメモリー

Below the table, a red arrow labeled 'Click' points to the MBean attribute 'JVMRuntimeMBean.HeapFreeCurrent'. A blue callout box contains the text 'MBean名、属性名を確認できる' (You can check the MBean name and attribute name). A blue arrow labeled '項目が対応' (Item corresponds) points from the table row to the '現在の空きヒープ' label in the main console content area.

# MBean をどのように呼び出すか

- MBean リファレンス
  - 引数や型、デフォルト値などを確認できる。

ヘルプ画面から

WEBLOGIC SERVER  
MBean REFERENCE

Contents View Bookmarks Bookmark Current Topic

- ⊕ JIStatisticsRuntimeMBean
- ⊕ JTATransactionStatisticsRuntimeMBean
- ⊖ **JVMSRuntimeMBean**
  - Attributes
  - Operations
- ⊕ KodoDataCacheRuntimeMBean
- ⊕ KodoPersistenceUnitRuntimeMBean
- ⊕ KodoQueryCacheRuntimeMBean
- ⊕ KodoQueryCompilationCacheRuntimeMBean
- ⊕ LibraryRuntimeMBean
- ⊕ LogBroadcasterRuntimeMBean
- ⊕ LogRuntimeMBean
- ⊕ MailSessionRuntimeMBean
- ⊕ MANASyncReplicationRuntimeMBean
- ⊕ MANReplicationRuntimeMBean

### HeapFreeCurrent

The current amount of memory (in bytes) that is a

Privileges	Read only
Type	long

### HeapFreePercent

Percentage of the maximum memory t

MBean 属性の型、操作の引数などの詳細情報

MBean の一覧

# MBean をどのように呼び出すか

該当の MBean を見つけたら:

どのMBean ツリーを参照するか把握する

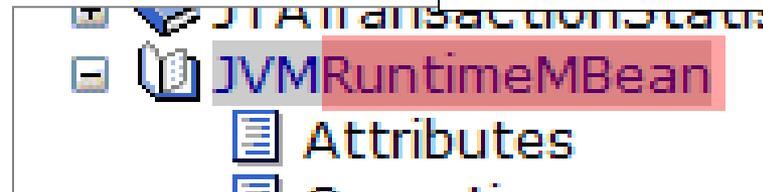
“RuntimeMBean” がつく

→ 実行時 MBean ツリー

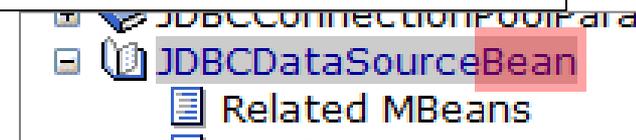
“RuntimeMBean” がつかない

→ コンフィグレーション MBean ツリー

実行時 MBean の例



コンフィグレーション MBean の例



serverRuntime() コマンドでサーバ実行時ツリーへ切り替える

接続完了  
後の状態

```
wls:/mydomain/serverConfig> serverRuntime()
```

実行例

ロケーションがserverRuntimeツリーに移動しました。これは、ServerRuntimeMBeanをルートとする読取り専用のツリーです。

# MBean をどのように呼び出すか

find() コマンドで MBean オブジェクトを検索

“MBean” は不要

```
wls:/mydomain/serverRuntime> find('JVMSRuntime')
Finding 'JVMSRuntime' in all registered MBean instances ...
/
JVMSRuntime
com.bea:ServerRuntime=AdminServer,Name=AdminServer,Type=JVMSRuntime。
```

実行例

getPath() コマンドでツリー上のパスを確認

```
wls:/mydomain/serverRuntime> getPath('com.bea:ServerRuntime=
AdminServer,Name=AdminServer,Type=JVMSRuntime')
'JVMSRuntime/AdminServer'
```

実行例

cd() コマンドでツリーを移動

```
wls:/mydomain/serverRuntime> cd('JVMSRuntime/AdminServer')
wls:/mydomain/serverRuntime/JVMSRuntime/AdminServer>
```

実行例

# MBean をどのように呼び出すか

get() コマンドや ls() コマンドで MBean の属性を取得

```
wls:/mydomain/serverRuntime/JVMRuntime/AdminServer>
```

```
get('HeapFreeCurrent')
```

```
152358560L
```

```
wls:/mydomain/serverRuntime/JVMRuntime/AdminServer> ls()
```

```
-r--    HeapFreeCurrent          152358560
-r--    HeapFreePercent          79
-r--    HeapSizeCurrent          259588096
-r--    HeapSizeMax              518979584
-r--    Type                      JVMRuntime
...
```

実行例

ヘルプ画面で確認した属性名

# 実際に作成したスクリプトの例

[simplemonitor.py]

```
from java.lang.Thread import *  
  
connect('username', 'password', 't3://localhost:7001')  
serverRuntime()  
cd('JVMRuntime/AdminServer')  
print('---- HeapFreeCurrent ---')  
while(1):  
    print(get('HeapFreeCurrent'))  
    Thread.sleep(60000)
```

WLST のスクリプトでは処理のブロックを  
インデントで表現する

スクリプトを実行し、終了  
java weblogic.WLST simplemonitor.py

# 事例紹介、デモ

- WebLogic Scripting Tool(WLST) とは？
- WLST でどのようなことができるか
- 必要な情報の探し方、WLST でのアクセス方法
- 事例紹介、デモ
  - WLST による監視: システムの監視、異常時のアクション
  - WLST によるサーバ制御: 確実なシャットダウン



- ① 監視項目に該当する MBean の確認
- ② 自動化
  - ・ WLST スクリプトモードの利用
- ③ 高負荷時のアクション処理
  - A) ログ採取の方法
    - ・ WLS のリソース情報
    - ・ スレッドダンプ
    - ・ OS 情報／ヒープダンプ
    - ・ 業務アプリケーション情報
  - B) 通知方法
    - ・ E-Mail での管理者への通知
    - ・ 作り込みの必要性

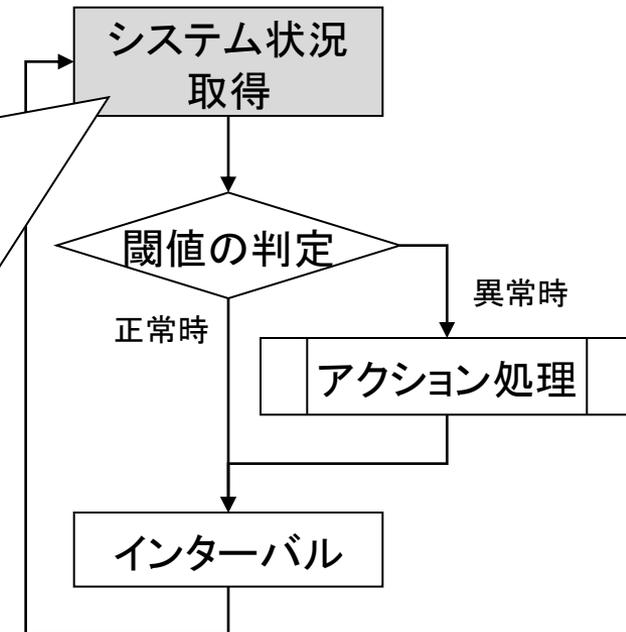
## JVM メモリ状況 (HeapSize) を監視する例

システム内の WLS サーバ名を取得するには？

```
domainConfig()  
svrNames = cmo.getServers()
```

サーバで現在のヒープ使用量を取得するには？

```
domainRuntime()  
for svr in svrNames:  
    cd (“/ServerRuntimes/” + svr.getName() +  
        “/JVMRuntime/” + svr.getName())  
    Size = cmo.getHeapSizeCurrent()  
    Free = cmo.getHeapFreeCurrent()  
    Used = Size - Free
```



# 運用事例1 (監視)

# —WLSTによる自動化—

閾値設定ファイルからの閾値の読み込み

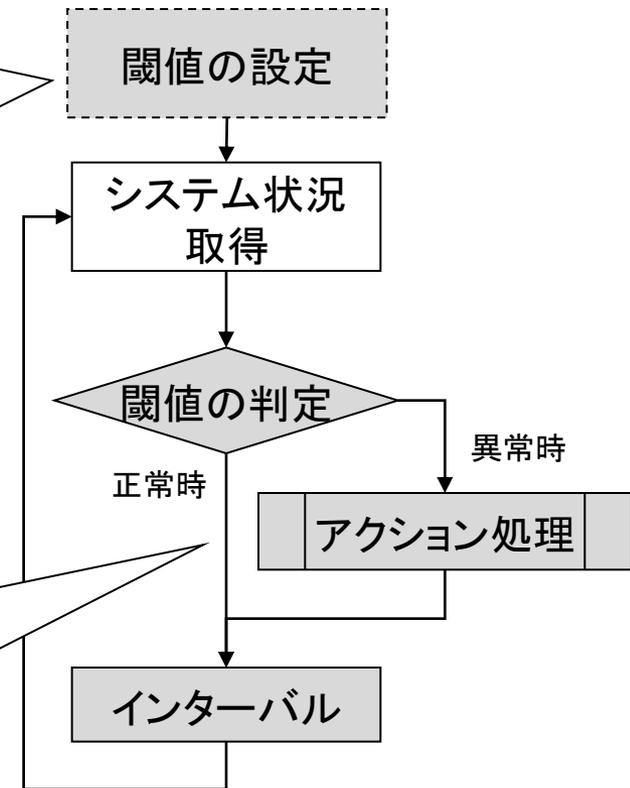
```
threshold = [] # 閾値格納変数
HEAP=1 # ヒープ情報 INDEX
def Init(inpfile): # 初期化関数
    inp = open(inpfile)
    threshold.append(line)
```

※inpfileは、各モニタ用の閾値設定ファイル

閾値設定と Used を判定し、アクションを実行

```
if Used > threshold [HEAP]:
    ActionMethod # アクション処理実行
    Thread.sleep(インターバル)
```

※インターバル処理 (Thread.sleep) では、javaSEのメソッドを使用するため  
WLSTのpyファイルの先頭で『from java.lang.Thread import \*』の記述が必要



# 運用事例1 (監視)

# 一ログ採取方法一

各サーバのヒープダンプ情報を採取します。  
ヒープダンプは javaVM の機能を利用するため事前準備が必要です。

## 事前準備

- ・WLSの起動時オプション (javaVMの引数) に下記を追加する  
『 `-Xrunhprof:heap=all,cutoff=0,doe=n` 』
- ・監視ツールが WLS サーバと別のマシン上にある場合は、ssh や rsh の設定をしておく必要がある

## 採取処理

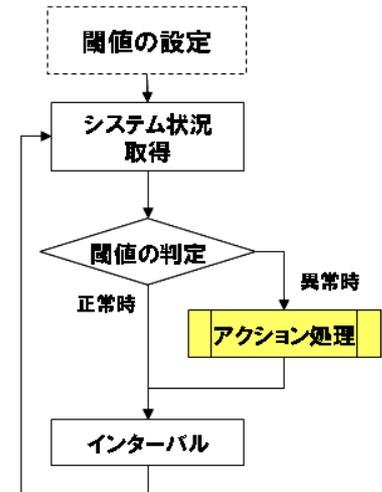
- ・WLS サーバのIPアドレスを取得する (監視ツールがリモートマシンの場合)

```
domainRuntime()  
cd("/ServerRuntimes/ ServerName")  
serverAddress = cmo.getListenAddress()
```

- ・サーバのヒープダンプ情報の取得

```
os.system(" ssh ユーザ名@" + serverAddress +  
"kill -3 `ps -ef|grep AdminServer | grep -v grep | grep weblogic.Server | awk '{print $2}'`")
```

osのkillコマンドでWLSのjava  
プロセスに対しSIGQUITを  
送信し、ヒープダンプを出力  
させる



# 運用事例1 (監視)

# —ログ採取方法—

各サーバのスレッドダンプ情報を採取します。  
スレッドダンプ情報の採取は WLSTコマンドでおこないます。

## 採取処理

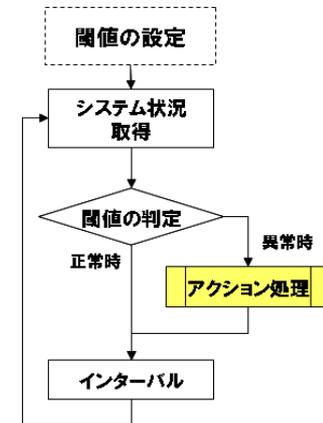
- ・スレッドダンプを採取する

```
threadDump(fileName='threadDumpfile',serverName='svrname')
```

(参考) 標準出力には出さず、  
ファイルにのみ出力する場合

```
cd (" /ServerRuntimes/" + svrname + " /JVMRuntime/" + svrname)  
dump = cmo.getThreadStackDump()  
print(str(dump))
```

スレッドダンプ情報の出力先ファイル  
デフォルトでカレントディレクトリに  
Thread\_Dump\_adminServer.txt が作成される



# 運用事例1 (監視)

# —通知方法—

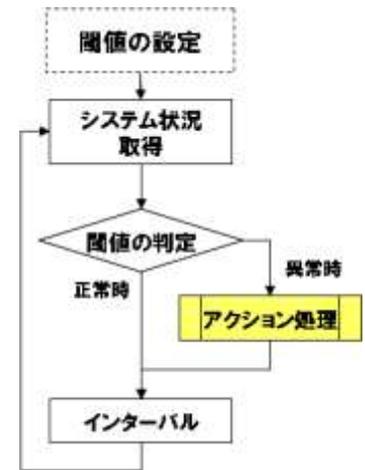
E-mailでの通知処理は独自で実装します。  
例は、Python の『smtplib』ライブラリを利用しています。  
NoticeActionメソッドを呼出し通知処理を実行します。

```
def NoticeAction():
```

```
    To = 'To: SystemAdmin@xxx.co.jp'  
    From = 'From: MontorTool@yyy.co.jp'  
    Subj = 'Subject: Simple test message ¥n '  
    Msg = 'Exceeded the threshold of the memory'  
    mdata = To + '¥n' + Subj + Msg
```

```
    server = smtplib.SMTP('メールサーバ')  
    server.login(login , password)
```

```
    server.sendmail(From, [To], mdata)
```



通知先と通知メッセージ  
の作成

メールサーバに接続

通知文を送信

## 事例2(制御)

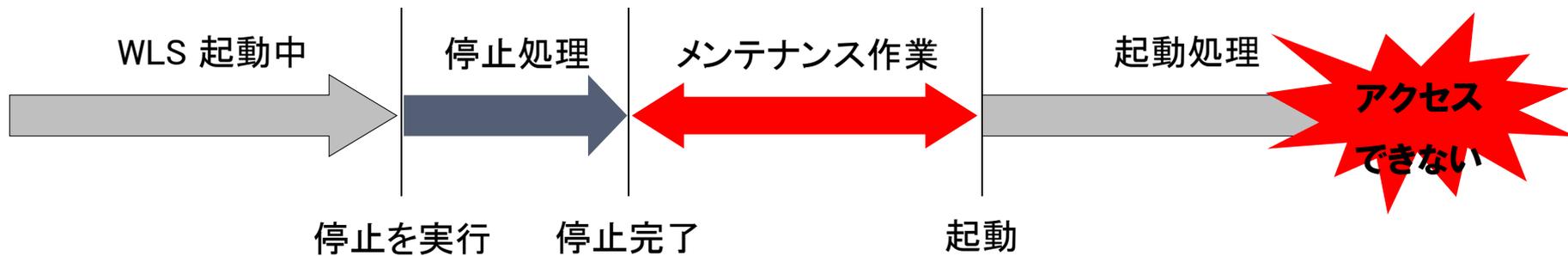
## —障害—

夜間にサーバの再起動を行っているが、就業時刻になってもシステムにアクセスできなかった。

### <<背景>>

システムのメンテナンス作業のため、日時バッチ処理にて以下の処理を実施している。

- ・バッチ処理で WLS を停止
- ・システムのメンテナンスを実施
- ・WLS を再起動



## 事例2 (制御)

## —原因—

[サーバをログを遡ってみて見ると…]

```
<BEA-141281> <unable to get file lock, will retry ...>  
(略)
```

起動時の標準出力

```
<BEA-000383> <クリティカルなサービスが失敗しました。サーバーは停止されます>  
<BEA-000365> <サーバー状態がFORCE_SHUTTING_DOWNに変化しました>
```

起動途中で既にサーバが起動しているメッセージ (BEA-141281) が出力されていた。  
その後、起動処理が強制終了されていた。

```
<BEA-002638> <. . . Graceful shutdownがリモートで発行されました>  
(略)
```

停止時のサーバログ

```
<BEA-101275> <. . . レプリケートされないセッションを検出しました. . . >  
<BEA-101276> <中断開始の1分後、. . . 待機します。>
```

WLSに有効なHTTPセッションが存在し、停止処理が中断していた。

- 実行中の処理の終了を待ち合わせていたため、WLSが停止していなかった
- その結果、WLSが二重起動の状況となり再起動に失敗した
- 停止途中のWLSは、HTTPセッションがタイムアウト後に正常終了するため、就業時刻には、システム内で起動しているサーバが存在していなかった

### 【業務への影響】

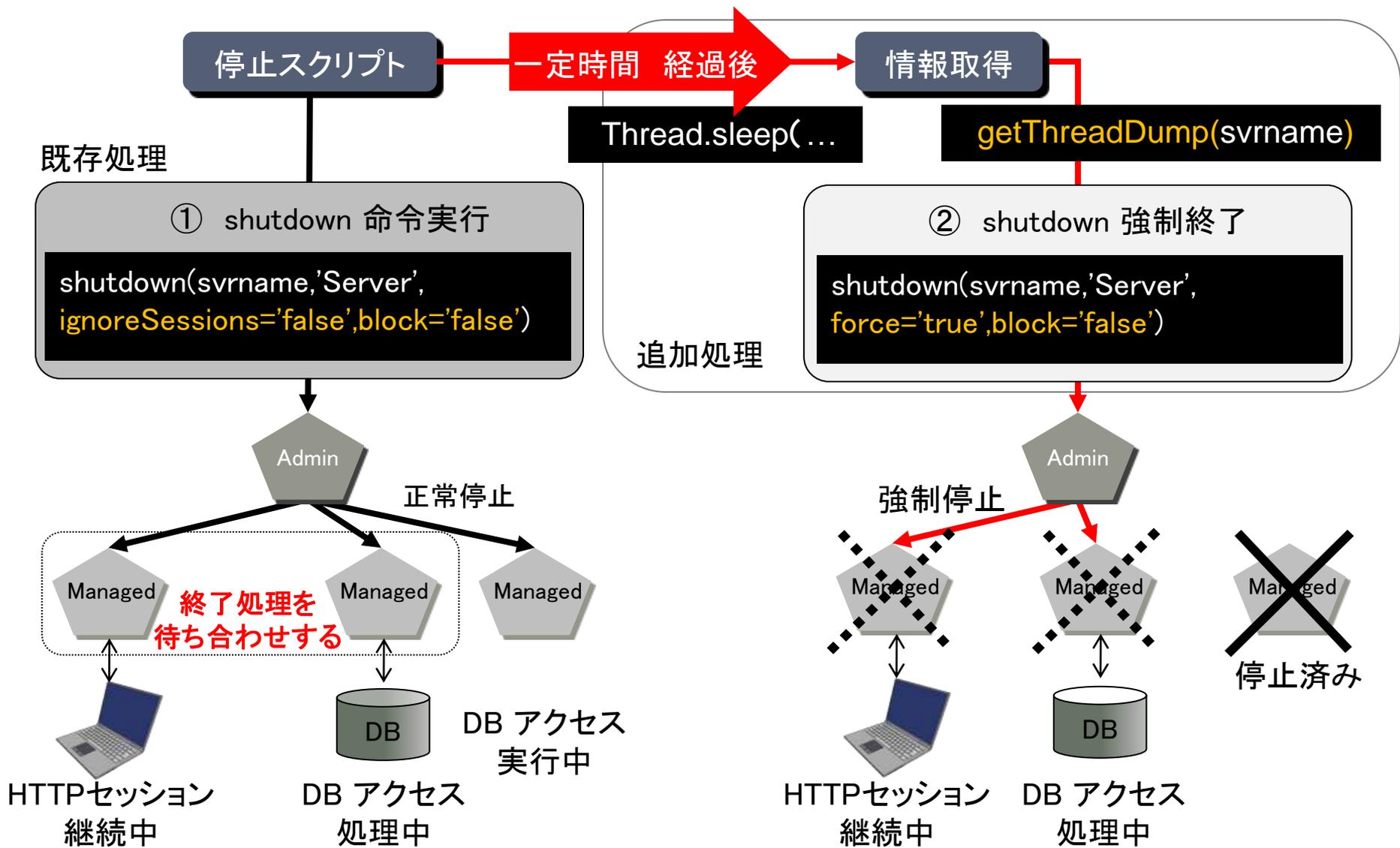
- 実行中の業務トランザクションを強制終了できるか？
- 有効な HTTP セッションを破棄できるか？
- 起動時刻の制限はあるか？

### 【その他 注意点】

- 強制終了した理由を確認するための情報取得
- データベースから WLS に応答を返さない場合の対応  
(DB マシンダウン、ネットワーク切断、SQL 処理遅延など)

# 運用事例2(制御)

# —実行イメージ—



# 概要

- WebLogic Scripting Tool(WLST) とは？
- WLST でどのようなことができるか
- 必要な情報の探し方、WLST でのアクセス方法
- 事例紹介
- デモ
  - 事例1. WLST による監視:ヒープサイズの監視

# 運用事例1 (監視)

# —MBean確認方法—

例:現在の使用ヒープサイズ

② 管理コンソール画面右上のヘルプをクリックし、管理コンソールのヘルプを表示

① 確認したい値を管理コンソールのモニタ等で確認して表示します。

現在のヒープ サイズ: 532742144

現在のヒープ サイズ: 532742144

MBean 属性:  
JVMRuntimeMBean.HeapSizeCurrent

現在のヒープサイズ JVMヒープの現在のサイズ (バイト)。

MBean 属性:  
JVMRuntimeMBean.HeapSizeCurrent

③ ヘルプ画面で該当項目を確認します。

[現在の使用ヒープ] の値は **JVMRuntimeMBean** で保持しており、属性の名称は **HeapSizeCurrent** であることが判ります。

# 運用事例1 (監視)

# —MBean確認方法—

④ WLST を起動し、実行時 MBean 階層に移動して find() コマンドで JVMRuntime を検索します。

```
wls:/mydomain/serverConfig> serverRuntime()
```

```
ロケーションが serverRuntime ツリーに移動しました。  
ルートとする読み込み専用のツリーです。  
詳細については、help(serverRuntime) を使用してください
```

実行時 MBean 階層に移動する。

```
wls:/mydomain/serverRuntime> find('JVMRuntime')  
Finding 'JVMRuntime' in all registered MBean instances ...  
/  
JVMRuntime  
com.bea:ServerRuntime=AdminServer,Name=AdminServer  
time
```

検索結果が “/” なので、**JVMRuntimeMBean** の MBean インスタンスは**実行時 Mbean (ServerRuntimeMBean)階層の直下に存在している**ことが判る

⑤ JVMRuntime に移動します。

```
wls:/mydomain/serverRuntime> cd('JVMRuntime')  
wls:/mydomain/serverRuntime/JVMRuntime> ls()  
dr-- AdminServer  
  
wls:/mydomain/serverRuntime/JVMRuntime> cd('AdminServer')
```

cd() でJVMRuntime に移動して ls() で内容を確認すると、AdminServer という名称の JVMRuntime インスタンスが存在しているので、cd() でこのインスタンスの中に移動する。

⑥ JVMRuntime/AdminServer に移動したら、ls() を実行して HeapSizeCurrent が存在するかを確認します。

```
wls:/mydomain/serverRuntime/JVMRuntime/AdminServer> ls()  
-r-- HeapFreeCurrent 488842368  
-r-- HeapFreePercent 91  
-r-- HeapSizeCurrent 532742144
```

ls() を実行すると、HeapSizeCurrent が存在していることを確認できる。

# 補足資料

- Python モジュールの利用
- TIPS (WLST コマンド)
- サンプルスクリプト

# 補足資料1 – Python モジュールの利用

## 文字列操作

```
> name = ('taro', 'jiro')
> print 'Hello %s and %s' % name
```

## ウェイト

```
> import time
> time.sleep(5)
```

## ファイルの操作

```
> f = open('c:/temp/test.txt', 'w')
> f.write("hello¥n")
> f.close()
```

## 入力を求める

```
➤ name = raw_input('your name? ')
```

## ループ

```
> for i in range(1,10):
>     print i
```

## 現在の時間

```
> import time
> time.strftime('%Y-%m-%d %H:%M:%S')
```

## 文字列の変換

```
> import string
> string.atoi("5")
> str(5)
```

## コマンドを確認

```
> dir(time)
```

# 補足資料2 - TIPS

prompt()

プロンプトの表示／非表示を切り替える

jndi()

JNDIツリーを表示するモード

help()

コマンドおよび変数に関する情報を表示

easeSyntax() コマンド（非サポート）

一部のコマンドを簡易入力することが可能

```
wls:/mydomain/serverConfig> easeSyntax()
```

```
wls:/mydomain/serverConfig> pwd
```

```
'serverConfig:/'
```

```
wls:/mydomain/serverConfig> cd Servers
```

```
wls:/mydomain/serverConfig/Servers> ls
```

```
....
```

# 補足資料3 - サンプルスクリプト - 事例1

## ■ ■ 事前準備 ■ ■

- 1)管理サーバと管理対象サーバを作成する
- 2)データソース(TstDrc)を作成する  
→対象は管理サーバと管理対象サーバの両方

### 設定ファイル(monitor\_conf)

```
-----  
weblogic          # 0 :WLS Login Id      (USER)  
welcome1         # 1 :WLS Login Password  (PWD)  
localhost:7111   # 2 :WLS Adimin URL      (WLS_URL)  
To:username@oracle.com # 3 :mail to address  (TO_ADDR)  
From: username@oracle.com # 4 :mail from address (FROM_ADDR)  
10.XXX.XXX.XXX  # 5 :mail server IP     (SMTP_SERVER)  
smtp_login_user  # 6 :SMTP Login Id     (SMTP_USER)  
smtp_login_passwd # 7 :SMTP Login Password (SMTP_PWD)  
1                # 8 :HEEPDUMP_LOCAL_FLAG (LFLAG) #1:local, 0:remote  
0                # 9 :MAIL_FLAG         (MFLAG)   #1:send, 0:no send  
TstDrc          # 10:JDBC DataSource Name (DATASRC_NAME)  
10              # 11:Interval         (INTERVAL)  
C:\jdk160_24\bin # 12:JAVA HOME + \bin (JAVA_HOME) #Windows  
C:\PsTools      # 13:PSEXEC_HOME      (PSEXEC_HOME) #Windows  
remote_login_user # 14:REMOTE Login Id  (REMOTE_USER) #Windows  
remote_login_passwd # 15:REMOTE Login Password (REMOTE_PWD) #Windows  
D:\             # 16:REMOTE jmap.cmd HOME (REMOTE_JMAP_HOME) #Windows  
ssh_login_user  # 17:SSH Login Password (SSH_USER) #Linux/Unix  
512000000      # 18:HeapSize Limit  
2              # 19:JDBC Pool Session Limit  
5              # 20:Number Of Thread Limit
```

### スクリプトファイル(monitor.py)

```
-----  
import sys  
import datetime  
import smtplib  
from java.lang.Thread import *  
  
threshold = []  
serverNames = []  
  
USER      = ""  
PWD       = ""  
WLS_URL   = ""  
TO_ADDR   = ""  
FROM_ADDR = ""  
SMTP_SERVER = ""  
SMTP_USER = ""  
SMTP_PWD  = ""  
LFLAG     = 1  
MFLAG     = 0  
DATASRC_NAME = ""  
INTERVAL  = 5 * 1000 #5秒  
JAVA_HOME = ""  
PSEXEC_HOME = ""  
REMOTE_USER = ""  
REMOTE_PWD  = ""  
REMOTE_JMAP_HOME = ""  
SSH_USER    = ""  
HEAP        = 0  
JDBC        = 1  
THREAD      = 2
```

# 補足資料3 – サンプルスクリプト – 事例1

スクリプトファイル(monitor.py) – 続き(2)

```
#####  
# 初期化関数  
#####  
def Init(conf):  
    inp = open(conf)  
    lines = inp.readlines()  
    global USER  
    global PWD  
    global WLS_URL  
    global TO_ADDR  
    global FROM_ADDR  
    global SMTP_SERVER  
    global SMTP_USER  
    global SMTP_PWD  
    global LFLAG  
    global MFLAG  
    global DATASRC_NAME  
    global INTERVAL  
    global JAVA_HOME  
    global PSEXEC_HOME  
    global REMOTE_USER  
    global REMOTE_PWD  
    global REMOTE_JMAP_HOME  
    global SSH_USER
```

スクリプトファイル(monitor.py) – 続き(3)

```
USER      = CutComment(lines[0])  
PWD       = CutComment(lines[1])  
WLS_URL   = CutComment(lines[2])  
TO_ADDR   = CutComment(lines[3])  
FROM_ADDR = CutComment(lines[4])  
SMTP_SERVER = CutComment(lines[5])  
SMTP_USER = CutComment(lines[6])  
SMTP_PWD  = CutComment(lines[7])  
LFLAG     = CutComment(lines[8])  
MFLAG     = CutComment(lines[9])  
DATASRC_NAME = CutComment(lines[10])  
INTERVAL  = CutComment(lines[11])  
JAVA_HOME = CutComment(lines[12])  
PSEXEC_HOME = CutComment(lines[13])  
REMOTE_USER = CutComment(lines[14])  
REMOTE_PWD = CutComment(lines[15])  
REMOTE_JMAP_HOME= CutComment(lines[16])  
SSH_USER  = CutComment(lines[17])  
  
for line in lines[18]:  
    threshold.append(CutComment(line))
```

# 補足資料3 – サンプルスクリプト – 事例1

スクリプトファイル(monitor.py) – 続き(4)

```
#####
# コメント& ¥nの削除関数
#####
def CutComment(line):
    idx = line.find('#')
    if idx > 0 :
        line = line[0:idx]

    return(line.rstrip())

#####
# 各WebLogicサーバの名前取得関数
#####
def getServerNames():
    domainConfig()
    return(cmo.getServers())

#####
# 各WebLogicサーバのListenAddress取得関数
#####
def getListenAddress(svrname):
    cd ("/ServerRuntimes/" + svrname)
    ip = cmo.getListenAddress()
    tmpstr = ip.split('/')
    if len(tmpstr) > 1:
        ip = tmpstr[1]
    return(ip)
```

スクリプトファイル(monitor.py) – 続き(5)

```
#####
# モニタ関数
#####
def Monitor():
    try:
        connect(USER,PWD,WLS_URL)
    except: #管理サーバへ接続失敗
        print("■■■■■Adomin WLS(" + WLS_URL + ") cannot Connect■■■■■")
        exit()

serverNames = getServerNames()
domainRuntime()

while(1): #モニタ開始
    d = datetime.datetime.today()
    print "[%s/%s/%s" % (d.year,d.month,d.day),
    print "%s:%s:%s]" % (d.hour,d.minute,d.second)
    print("ServerName          HeapSize          Thread
          JDBCPOOL")

    for svr in serverNames:
        print svr.getName() + '          ',
        CheckHeap(svr.getName())
        CheckThreadPool(svr.getName())
        CheckJDBCPOOL(svr.getName())
        #CheckJMS(svr.getName())
    print("-----")

    Thread.sleep(int(INTERVAL)*1000) #サンプリング間隔
```

# 補足資料3 – サンプルスクリプト – 事例1

スクリプトファイル(monitor.py) – 続き(6)

```
#####
# ヒープメモリチェック関数
#####
def CheckHeap(svrname):
    cd( "/ServerRuntimes/" + svrname + "/JVMSRuntime/" + svrname)
    Size = cmo.getHeapSizeCurrent()
    Free = cmo.getHeapFreeCurrent()
    Used = Size - Free

    print(str(Used) + "/" + str(threshold[HEAP]) + '      '),

    if Used > long(threshold[HEAP]):
        ActHeapDump(svrname)

#####
# スレッドプールチェック関数
#####
def CheckThreadPool(svrname):
    cd( "/ServerRuntimes/" + svrname + "/ThreadPoolRuntime/ThreadPoolRuntime" )
    Cnt1 = cmo.getExecuteThreadTotalCount()
    Cnt2 = cmo.getStandbyThreadCount()
    Cnt = Cnt1 - Cnt2
    print(str(Cnt) + "/" + str(threshold[THREAD]) + '      '),

    if Cnt > long(threshold[THREAD]):
        ActThreadDump(svrname)
```

スクリプトファイル(monitor.py) – 続き(7)

```
#####
# JDBCチェック関数
#####
def CheckJDBCPOOL(svrname):
    cd( "/ServerRuntimes/" + svrname + "/JDBCServiceRuntime/" + svrname +
        "/JDBCDataSourceRuntimeMBeans/" + DATASRC_NAME )
    Cnt = cmo.getActiveConnectionsCurrentCount()

    print(str(Cnt) + "/" + str(threshold[JDBC]) + '      '),

    if Cnt > long(threshold[JDBC]):
        ActThreadDump(svrname)

#####
# JMSキューチェック関数
#####
def CheckJMS(svrname):
    # ・サーバ インスタンス名 : AdminServer
    # ・JMS サーバ名 : MyJMSServer
    # ・JMS システム モジュール名 : MyJMSSystemModule
    # ・キュー名 : MyJMSQueue
    # を前提とする。

    cd( "/ServerRuntimes/AdminServer/JMSRuntime/AdminServer.jms/JMSServers/MyJMSServer/Destinations/MyJMSSystemModule/MyJMSQueue" )
    Cnt = cmo.getMessagesCurrentCount()
```

# 補足資料3 – サンプルスクリプト – 事例1

スクリプトファイル(monitor.py) – 続き(8)

```
#####
# 閾値越えのアクション関数 ヒープダンプ ヒープダンプをファイルに出力
#####
# Windows用のヒープダンプ採取アクション
def ActHeapDump(svrname):
    #ファイル名の設定
    d = datetime.datetime.today()
    dd = str(d.day) + str(d.hour) + str(d.minute) + str(d.second)
    filename = 'HeapDump_' + svrname + "_" + dd + '.bin'

    if int(LFLAG) > 0:
        # ローカルマシン上のWebLogicサーバに対してヒープダンプを採取する場合

        #プロセスIDの取得
        f = os.popen(JAVA_HOME + "jps -v | findstr " + svrname)
        pid = ".join(f)
        pid = pid[0:pid.find(' ')]

        #ヒープダンプ取得
        os.system(JAVA_HOME + "jmap -dump:format=b,file=" + filename + " " + pid + " > NUL
                2>&1")
    else:
        # リモートマシン上のWebLogicサーバに対してヒープダンプを採取する場合

        #IPアドレス取得
        ipaddr = getListenAddress(svrname)
```

スクリプトファイル(monitor.py) – 続き(9)

```
#####
#プロセスIDの取得
f = os.popen(JAVA_HOME + "jps -v -m " + ipaddr + " | findstr " + svrname)
pid = ".join(f)
pid = pid[0:pid.find(' ')]

#ヒープダンプ取得
os.system(PSEXEC_HOME + "PsExec.exe ¥¥¥¥" + ipaddr + " -u " + REMOTE_USER + "
        -p " + REMOTE_PWD + " -d " + REMOTE_JMAP_HOME + "jmap.cmd " +
        REMOTE_JMAP_HOME + " " + filename + " " + pid + " > NUL 2>&1")
NoticeAction("HEAP",svrname)

# Linux/Unix用のヒープダンプ採取アクション
#def ActHeapDump(svrname):
# if int(LFLAG) > 0:
# # ローカルマシン上のWebLogicサーバに対してヒープダンプを採取する場合
#
# #ヒープダンプ取得
# os.system("kill -3 `ps -ef | grep " + svrname + " | grep -v grep | grep weblogic.Server |
        awk '{print $2}'`)
# else:
# # リモートマシン上のWebLogicサーバに対してヒープダンプを採取する場合
#
# #IPアドレス取得
# ipaddr = getListenAddress(svrname)
#
# #ヒープダンプ取得
# os.system("ssh " + SSH_USER + "@" + ipaddr + " kill -3 `ps -ef | grep " + svrname + " |
        grep -v grep | grep weblogic.Server | awk '{print $2}'`)
# NoticeAction("HEAP",svrname)
```

# 補足資料3 – サンプルスクリプト – 事例1

スクリプトファイル(monitor.py) – 続き(10)

```
#####
# 閾値越えのアクション関数 スレッドダンプ スレッドダンプをファイルに出力
#####
def ActThreadDump(svrname):

#標準出力のリダイレクト
ofd = open('ThreadDump_' + svrname + '.log','a')
sys.stdout = ofd
sys.stderr = ofd

d = datetime.datetime.today()
print('[%s/%s/%s ' % (d.year,d.month,d.day)),
print('%s:%s:%s]' % (d.hour,d.minute,d.second))

cd (" /ServerRuntimes/" + svrname + " /JVMRuntime/" + svrname)
dump = cmo.getThreadStackDump()
print(str(dump))

#標準出力を戻す
sys.stdout = sys._stdout_
sys.stderr = sys._stderr_

NoticeAction("THREAD",svrname)
```

スクリプトファイル(monitor.py) – 続き(11)

```
#####
# 通知関数
#####
def NoticeAction(TYPE,svrname):
if int(MFLAG) > 0:
To = TO_ADDR
From = FROM_ADDR
Subj = 'Subject: Simple test message ¥n'
msg = 'Test Mail Body ' + svrname + ":" + TYPE + 'is Limit over'
Ctype= 'Content-Type: text/plain; charset=iso-2022-jp¥n'
mdata = To + '¥n' + 'Subject: ' + Subj + '¥n' + Ctype + msg

server = smtplib.SMTP(SMTP_SERVER)
#server.set_debuglevel(True)
#server.ehlo()
server.login(SMTP_USER,SMTP_PWD)
server.sendmail(From,[To], mdata)

#####
# メイン関数
#####
if __name__ == "main":
Init(sys.argv[1])
Monitor()
```

# 補足資料3 – サンプルスクリプト – 事例2

## ■ ■事前準備■ ■

1)管理サーバと管理対象サーバを作成する

設定ファイル(shutdown\_conf)

```
-----
weblogic      # 0 :WLS Login Id      (USER)
welcome1     # 1 :WLS Login Password  (PASSWD)
localhost:7111 # 2 :WLS Adimin URL      (WLS_URL)
30           # 3: Interval      (INTERVAL)
```

スクリプトファイル(shutdown.py)

```
-----
import datetime
from java.lang.Thread import *

infos      = []

USER       = ""
PWD        = ""
WLS_URL    = ""
INTERVAL   = 5 * 1000 #5秒

#####
# サーバ情報格納クラス
#####
class ServerInfo:
    def __init__(self,svrname,ip,port,flag):
        self.name = svrname
        self.ip   = ip
        self.port = port
        self.adminflag = flag
```

スクリプトファイル(shutdown.py) – 続き(2)

```
-----
#####
# 初期化関数
#####
def Init(conf):
    inp = open(conf)
    lines = inp.readlines()
    global USER
    global PWD
    global WLS_URL
    global INTERVAL

    USER      = CutComment(lines[0])
    PWD       = CutComment(lines[1])
    WLS_URL    = CutComment(lines[2])
    INTERVAL  = CutComment(lines[3])

#####
# コメント& ¥nの削除関数
#####
def CutComment(line):
    idx = line.find('#')
    if idx > 0 :
        line = line[0:idx]
    return(line.rstrip())
```

# 補足資料3 – サンプルスクリプト – 事例2

スクリプトファイル(shutdown.py) – 続き(3)

```
#####
# 各WebLogicサーバの名前取得関数
#####
def getServerNames():
    domainConfig()
    return(cmo.getServers())

#####
# 各WebLogicサーバのListenAddress取得関数
#####
def getListenAddress(svrname):
    cd("/ServerRuntimes/" + svrname)
    ip = cmo.getListenAddress()
    tmpstr = ip.split('/')
    if len(tmpstr) > 1:
        ip = tmpstr[1]
    return(ip)

#####
# 各WebLogicサーバのListenPort取得関数
#####
def getListenPort(svrname):
    cd("/ServerRuntimes/" + svrname)
    port = cmo.getListenPort()
    return(port)
```

スクリプトファイル(shutdown.py) – 続き(4)

```
#####
# WebLogicサーバが管理サーバか否かの情報取得関数
#####
def getAdminServerflag(svrname):
    flag = True
    cd("/ServerRuntimes/" + svrname)
    flag = get("AdminServer")
    return(flag)

#####
# 事前処理関数
#####
def PreShutDown():
    serverNames = getServerNames()
    domainRuntime()
    for svr in serverNames:
        svrname = svr.getName()
        ip = getListenAddress(svrname)
        port = getListenPort(svrname)
        flag = getAdminServerflag(svrname)
        infos.append(ServerInfo(svrname,ip,port,flag))
```

# 補足資料3 – サンプルスクリプト – 事例2

スクリプトファイル(shutdown.py) – 続き(5)

```
#####
# 終了処理メイン関数
#####
def ShutDown():
    try:
        connect(USER,PWD,WLS_URL)
    except:
        #管理サーバへ接続失敗
        print("■■■■Adomin WLS(" + WLS_URL + ") cannot Connect■■■■")
        exit()

#事前準備
PreShutDown()

#
# フェーズ1 シャットダウン処理
#
for info in infos:
    if not info.adminflag:
        ShutDownPh1(info)

#
# 待機時間
#
Thread.sleep(int(INTERVAL)*1000)
```

スクリプトファイル(shutdown.py) – 続き(6)

```
#####
#
# フェーズ2 強制シャットダウン処理
#
for info in infos:
    if not info.adminflag:
        ShutDownPh2(info)
    else:
        admin_svrname = str(info.name)

#
# 管理サーバの停止
#
shutdown(admin_svrname,'Server',force='true',block='false')
```

# 補足資料3 – サンプルスクリプト – 事例2

スクリプトファイル(shutdown.py) – 続き(7)

```
#####
# 終了処理フェーズ1関数
#####
def ShutDownPh1(info):
    svrname = str(info.name)
    cd ("/ServerLifeCycleRuntimes/" + svrname)
    st = cmo.getState()

    if st == "SHUTDOWN":
        infos.remove(info)
    else:
        d = datetime.datetime.today()
        print("■■■■■[%s/%s/%s %s:%s:%s] " % (d.year,d.month,d.day,d.hour,d.minute,d.second)),
        print("START WLS(" + svrname + ") SHUTDOWN Phase1 ■■■■■")
    try:
        shutdown(svrname,'Server',ignoreSessions='false',block='false')
    except:
        print(svrname + "(STATE:" + st + ") FAILED SHUTDOWN Phase1")
        dumpStack()
```

スクリプトファイル(shutdown.py) – 続き(8)

```
#####
# 終了処理フェーズ2関数
#####
def ShutDownPh2(info):
    svrname = str(info.name)
    cd ("/ServerLifeCycleRuntimes/" + svrname)
    st = cmo.getState()

    if st == "SHUTDOWN":
        infos.remove(info)
    else:
        #
        # スレッドダンプ採取
        #
        getThreadDump(svrname)

        #
        # 強制シャットダウン開始
        #
        d = datetime.datetime.today()
        print("■■■■■[%s/%s/%s %s:%s:%s] " %
            (d.year,d.month,d.day,d.hour,d.minute,d.second)),
        print("START WLS(" + svrname + ") SHUTDOWN Phase2 ■■■■■")

    try:
        shutdown(svrname,'Server',force='true',block='false')
    except:
        print(svrname + "(STATE:" + st + ") FAILED SHUTDOWN Phase2")
        dumpStack()
```

# 補足資料3 – サンプルスクリプト – 事例2

スクリプトファイル(shutdown.py) – 続き(9)

```
#####  
# スレッドダンプ採取  
#####  
def getThreadDump(svrname):  
  
    #標準出力のリダイレクト  
    ofd = open('ThreadDump_' + svrname + '.log','a')  
    sys.stdout = ofd  
    sys.stderr = ofd  
  
    d = datetime.datetime.today()  
    print('[%s/%s/%s ' % (d.year,d.month,d.day)),  
    print('%s:%s:%s]' % (d.hour,d.minute,d.second))  
  
    cd (" /ServerRuntimes/" + svrname + " /JVMRuntime/" + svrname)  
    dump = cmo.getThreadStackDump()  
    print(str(dump))  
  
    #標準出力を戻す  
    sys.stdout = sys._stdout_  
    sys.stderr = sys._stderr_  
  
#####  
# メイン関数  
#####  
if __name__ == "main":  
    Init(sys.argv[1])  
    ShutDown()
```

# OTNセミナーオンデマンド

コンテンツに対する  
ご意見・ご感想を是非お寄せください。

OTNオンデマンド 感想



[http://blogs.oracle.com/oracle4engineer/entry/otn\\_ondemand\\_questionnaire](http://blogs.oracle.com/oracle4engineer/entry/otn_ondemand_questionnaire)

上記に簡単なアンケート入力フォームをご用意しております。

セミナー講師/資料作成者にフィードバックし、  
コンテンツのより一層の改善に役立てさせていただきます。

是非ご協力をよろしくお願いいたします。

# OTNセミナーオンデマンド

日本オラクルのエンジニアが作成したセミナー資料・動画ダウンロードサイト

## 掲載コンテンツカテゴリ(一部抜粋)

Database 基礎

Database 現場テクニック

Database スペシャリストが語る

Java

WebLogic Server/アプリケーション・グリッド

EPM/BI 技術情報

サーバー

ストレージ



超入門! Oracle データベースって何  
再生時間: 60分

100以上のコンテンツをログイン不要でダウンロードし放題

データベースからハードウェアまで充実のラインナップ

毎月、旬なトピックの新作コンテンツが続々登場

## 例えばこんな使い方

- 製品概要を効率的につかむ
- 基礎を体系的に学ぶ/学ばせる
- 時間や場所を選ばず(オンデマンド)に受講
- スマートフォンで通勤中にも受講可能



毎月チェック!



コンテンツ一覧 はこちら

<http://www.oracle.com/technetwork/jp/ondemand/index.html>

新作&おすすめコンテンツ情報 はこちら

<http://oracletech.jp/seminar/recommended/000073.html>

OTNオンデマンド



# オラクルエンジニア通信

オラクル製品に関わるエンジニアの方のための技術情報サイト

## オラクルエンジニア通信 - 技術資料、マニュアル、セミナー

Oracleエンジニアのための技術情報サイト by Oracle Japan

[新着情報を知りたい](#)

[技術資料を探したい](#)

[セミナーを受けたい](#)

### About

Oracleエンジニアの方がスキルアップしていただくために、厳選した情報をお届けしています

技術資料



インストールガイド・設定チュートリアルetc. 欲しい資料への最短ルート

アクセスランキング



他のエンジニアは何を見ているのか？人気資料のランキングは毎月更新

特集テーマ Pick UP



性能管理やチューニングなど月間テーマを掘り下げて詳細にご説明

技術コラム



SQLスクリプト、索引メンテナンスetc. 当たり前運用/機能が見違える!?

<http://blogs.oracle.com/oracle4engineer/>

オラクルエンジニア通信





The screenshot shows the top section of the oracletech.jp website. On the left is the 'oracletech.jp' logo with the tagline '好奇心が、エンジニア人生を豊かにする。'. On the right is the 'ORACLE' logo, a search bar, and social media icons for Twitter, Facebook, Ustream, YouTube, and RSS. Below these is a red navigation bar with five buttons: '製品/技術情報', 'スキルアップ', 'セミナー', 'キャンペーン', and 'ちょっと一息'.

製品/技術  
情報



Oracle Databaseっていくら？オプション機能も見積れる簡単ツールが大活躍

セミナー



基礎から最新技術までお勧めセミナーで自分にあった学習方法が見つかる

スキルアップ



ORACLE MASTER !  
試験頻出分野の模擬問題と解説を好評連載中

Viva!  
Developer



全国で活躍しているエンジニアにスポットライト。きらりと輝くスキルと視点を盗もう

<http://oracletech.jp/>

oracletech



あなたにいちばん近いオラクル



# Oracle Direct

まずはお問合せください

Oracle Direct



システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。  
システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

## Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。  
[http://www.oracle.co.jp/inq\\_pl/INQUIRY/quest?rid=28](http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28)

※フォームの入力にはログインが必要となります。  
※こちらから詳細確認のお電話を差し上げる場合がありますので  
ご登録の連絡先が最新のものになっているかご確認下さい。

## フリーダイヤル

0120-155-096

※月曜～金曜  
9:00～12:00、13:00～18:00  
(祝日および年末年始除く)

ORACLE

# **Hardware and Software Engineered to Work Together**

**ORACLE®**