ORACLE®

WebLogic Server Tuning und Tools

Michael Fuhr
Principal Sales Consultant
November, 2010

Agenda

- WebLogic Server Tuning
- WebLogic Server Ant Tasks
- WebLogic Server Entwickler Tools
- WebLogic Scripting Tool

WebLogic Server Tuning

Development Mode vs. Production Mode

- Development Mode
  - Verwendung von SUN Hotspot JDK
  - Demo Zertifikate für SSL werden verwendet
  - Auto deployment ist eingeschaltet
  - Server Instanzen rotieren die Log-Dateien während des Startens
  - Administration Server verwendet eine automatisch erzeugte boot.properties
  - Default Maximum Kapazität für JDBC Datasource ist 15
  - debugFlag für remote Debug sind eingeschaltet

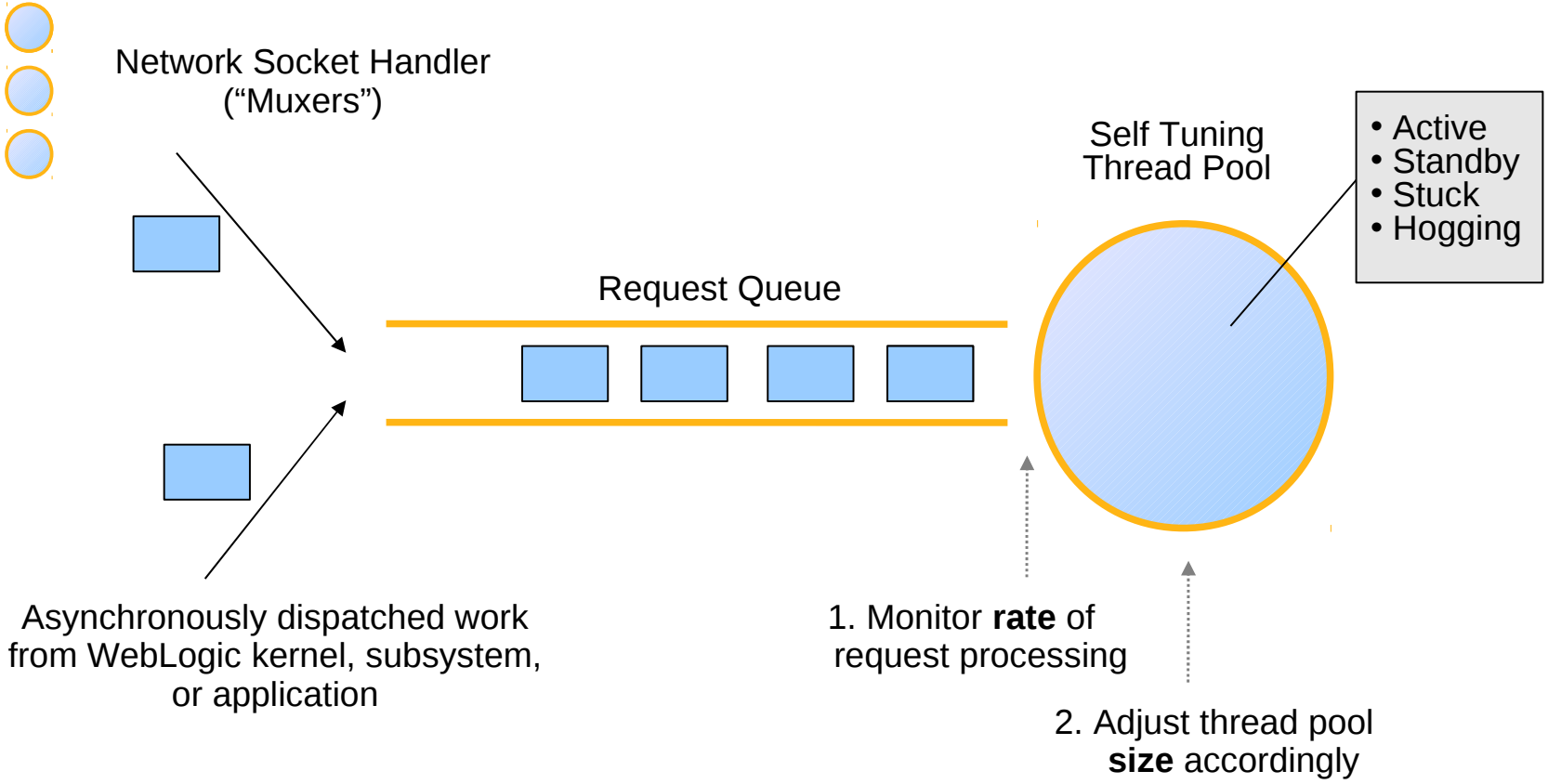
Development Mode vs. Production Mode

- Production Mode
  - JDK für Produktion Domain ist Oracle JRockit
  - Bei Verwendung von Demo Zertifikaten für SSL wird eine Warnung ausgegeben
  - Auto deployment ist ausgeschaltet
  - Server Instanzen rotieren ihre Log-Dateien bei einer Größe von 5MB
  - Administration Server erwartet Benutzername und Passwort während des Starts
  - Default Maximum Kapazität für JDBC Datasource ist 25
  - Die debugFlag für remote Debugging ist ausgeschaltet

Threads

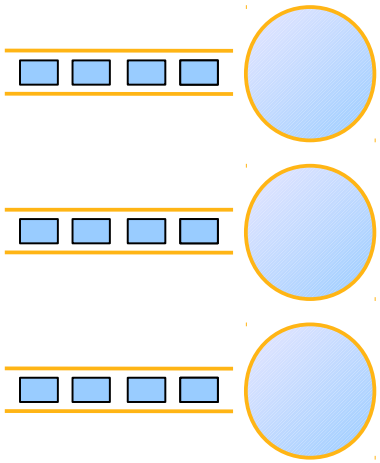
- Threads erhöhen Performanz und Funktionalität in unterschiedlichen Programmiersprachen
- Zwei Typen von Threads:
  - Native threads
    - Verwendet die Betriebssystem nativen Threads für Multithreaded Prozesse.
  - Green threads
    - Emuliert Multithreaded Umgebungen ohne Betriebssystem Funktionalitäten.

Work Manager
Der WebLogic Self Tuning Thread Pool



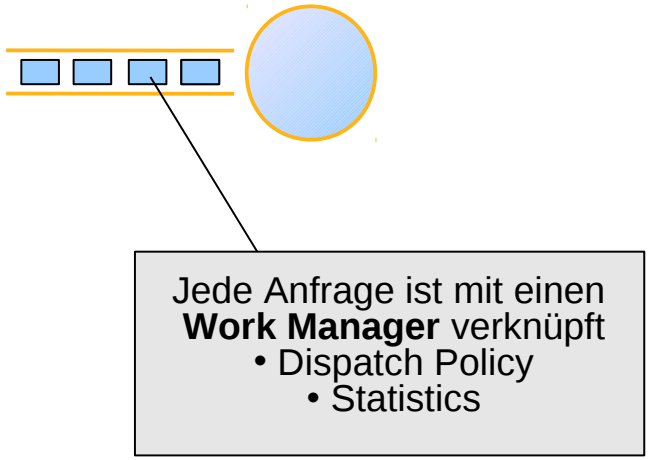
Work Manager

Alter Ansatz
viele unterschiedliche
Thread pools

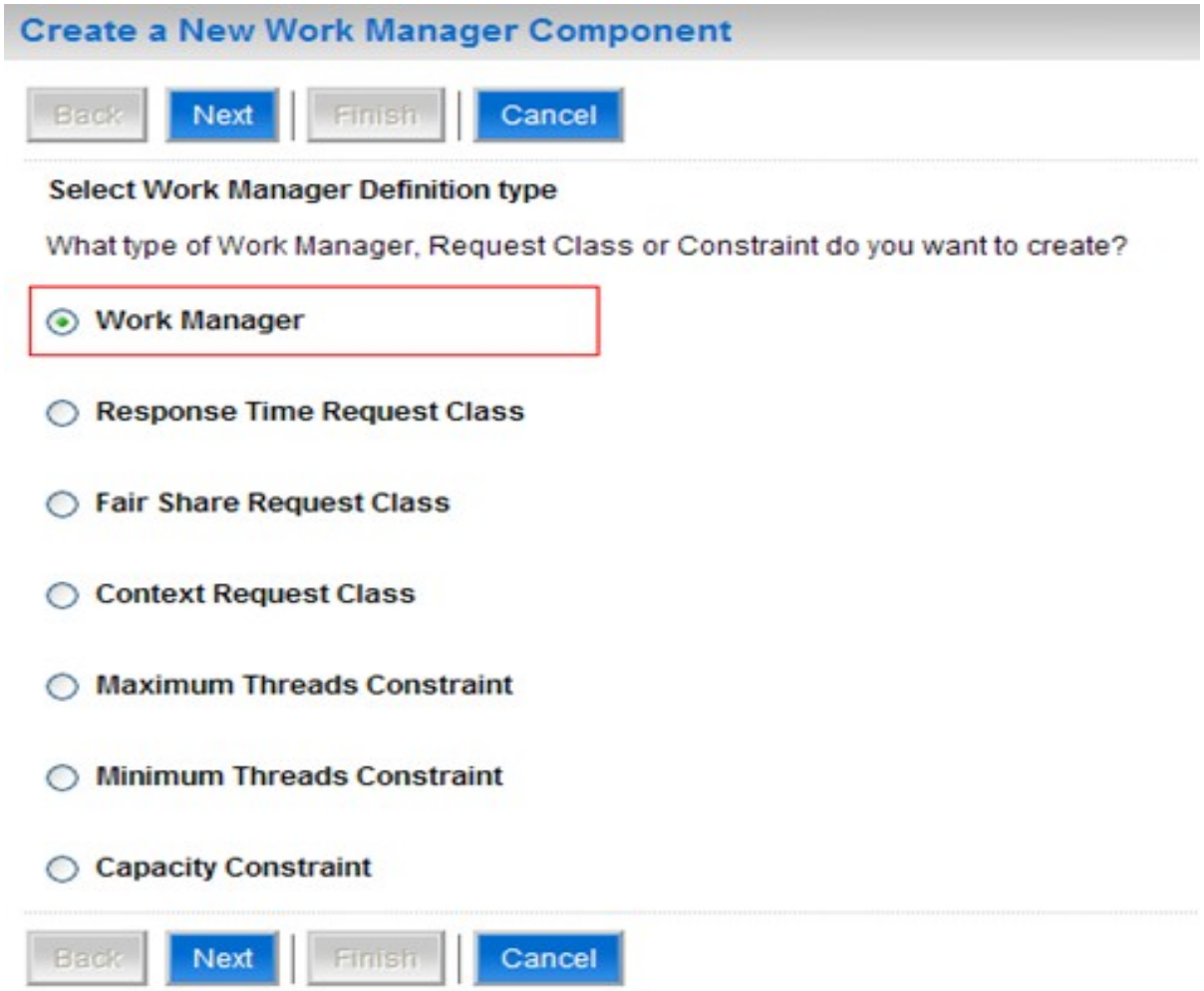


Neuer Ansatz

Priorisierte Request Queue

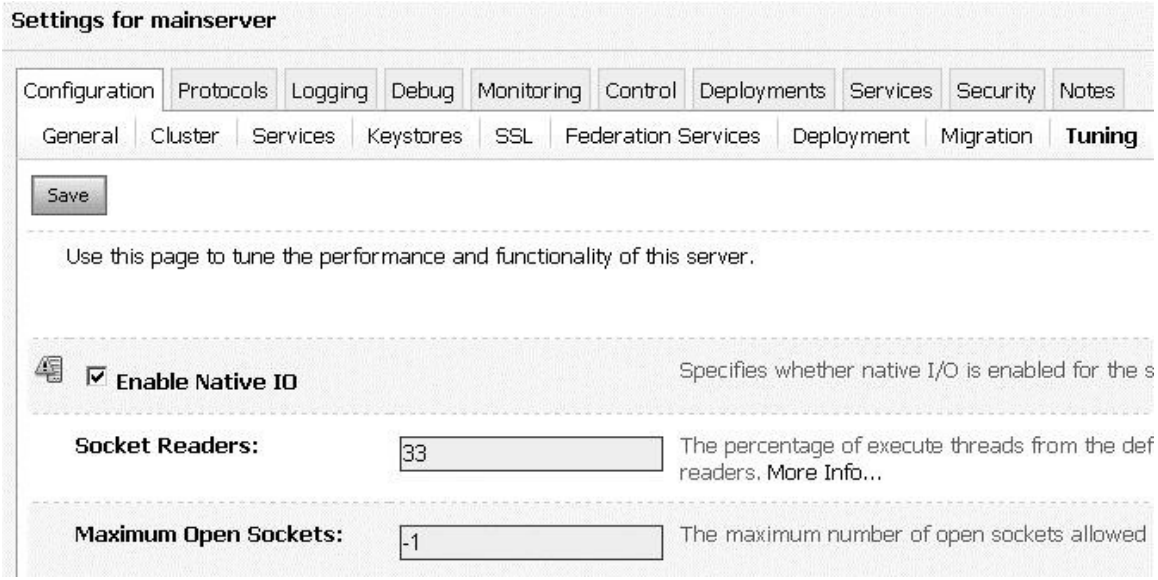


Work Manager



Native IO Performance Pack...

- Native Performance Packs verwenden einen Plattform-optimierten Socket Multiplexer.
- In der Administration Console kann Native IO eingeschaltet werden.



...Native IO Performance Pack

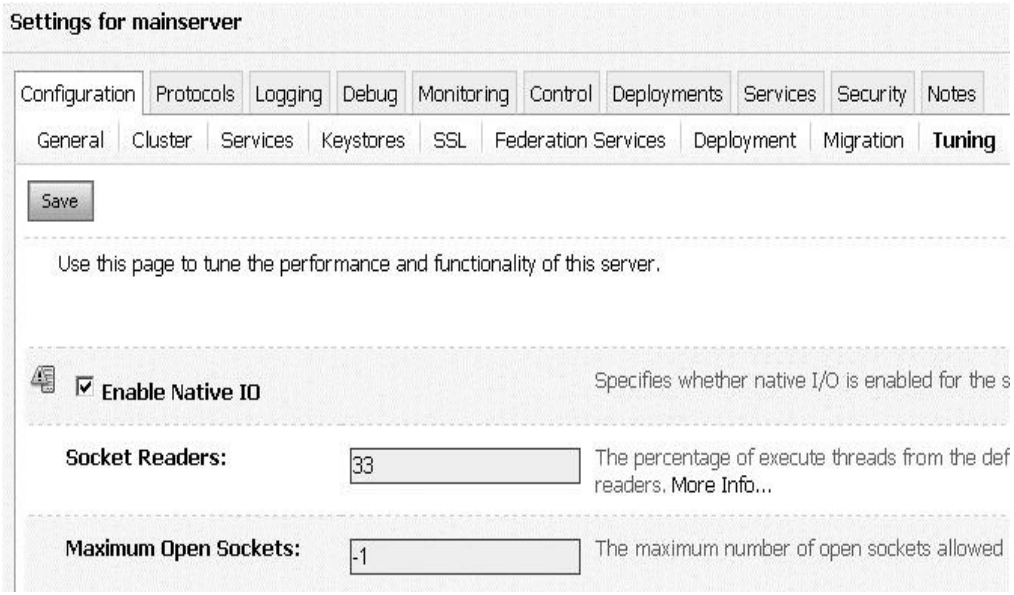
- Native Performance Packs sind standardmäßig aktiviert.
- Performance Packs sind Platfform abhängig.

Snippet: <config.xml>

```
<server>
<name>ServerName</name>
<listen-address>localhost</listen-address>
<native-io-enabled>true<native-io-enabled>
</server>
```

Socket Reader...

- Socket Readers sind die Prozentzahl von Threads innerhalb der Default Queue.
  - Default Wert: 33%
  - Minimal Wert: 1%
  - Maximum Wert: 99%
- Der optimale Wert ist Anwendungs-spezifisch



Administration Tools

- Configuration Wizard
  - GUI oder skriptbasiertes Tool zum Erzeugen und erweitern einer WebLogic Server Domain
  - Template basiert
- Administration Console
- Weblogic Scripting Tool (WLST)
  - Skript oder Kommandozeilen Tool für die gleiche Funktionalität in der Administration Console
- weblogic.Deployer
  - Kommandozeilen Tool für das Deployment

WebLogic Server Ant Tasks

WebLogic Server ant Tasks

- Einfache Integration von WebLogic Server Aufgaben in ein Ant Build Skript
- Task für starten, stoppen, restart von Server-Instanzen
  - <taskdef name="wlserver"
- Task für deploy, undeploy, redeploy
  - <taskdef name="wldeploy"
- Task für direktes Ausführen von WLST Skripten
  - <taskdef name="wlst"

Ant wlst

```
<wlst
   debug="true"
   failOnError="false"
   executeScriptBeforeFile="true"
   fileName="./domain/domain-install.py">
</wlst>
```

Ant wldeploy

```
<wldeploy action="deploy"
          verbose="true"
          debug="true"
          name="testAPP"
          upload="true"
          source="${basedir}/app/testAPP.ear"
          user="${deploy.user}"
          password="${deploy.password}"
          adminurl="${deploy.adminurl}"
          targets="${deploy.targets}"
 />
```

WebLogic Server Entwickler Tools

WebLogic Server Entwickler Tools

- WebLogic Server bietet einige hilfreiche Tools für Entwickler

Tool	Beschreibung
weblogic.Deployer	Kommandozeilen Deployment Tool
weblogic.PlanGenerator	Erzeugt einen Deployment Plan für eine Anwendung
weblogic.DDConverter	Konvertiert DD zur aktuellen Version
weblogic.marathon.ddinit.EarInit	Erzeugt EAR Deployment Descriptoren
weblogic.marathon.ddinit.WebInit	Erzeugt Web Deployment Descriptoren
weblogic.appc	Compiler für JSP, EJB, überprüft Deployment Descriptoren

WebLogic Scripting Tool

WebLogic Scripting Tool (WLST)

- Skript basiertes Tool für die Administration einer Domain (erzeugen, konfigurieren, verwalten, überwachen, deployen Anwendungen)
- Basiert auf Jython – Pure Java Implementation von Python
- Großartig für Automatisierung von wiederkehrenden Aufgaben

Interaktionsmodus

- **Interaktiv**
  - Eingabe eines Kommandos über Kommandozeile
  - Online Modus: Verbindung zu einer WLS Instanz besteht
- **Skript**
  - Textdatei mit einer .py Dateiendung
  - Jython Kommandos
  - Ausführen von einer Sequenz von WLST Kommandos ohne Eingabe
- **Embedded**
  - Einfügen des WLST Interpreter in Java Code
  - Ausführen von WLST Kommandos aus einem Java Programm heraus

Verbindungsarten

- **Offline**: analog zum Configuration Wizard
  - Verwendet das Offline Configuration Framework
  - Lese- und Schreibzugriffe auf die Konfigurationsdaten (config.xml) oder in einem Domain Template JAR
  - Vorgesehen zur Erstellung einer Domain oder Modifikation einer nicht gestarteten Domain

- **Online**: analog zur Administration Console
  - JMX Client
  - Interagiert mit den Server MBeans
  - Vorgesehen für Runtime Management Tool: Konfiguration, Management, Deployment, Überwachung

Starten von WLST

- Umgebung setzten:
  - setWLSEnv.sh/cmd – setzt den Pfad und den Classpath
- Aufrufen von WLST:
  - java weblogic.WLST
  - java weblogic.WLST c:\myscripts\myscript.py
- Startet im Offline mode
- Verbinden zu einer Domain:
  - `wls:/offline>`
    `connect('weblogic','weblogic','localhost:7001')`

Verfügbare MBean

- **domainConfig**
  - Domain Konfiguration MBeans Hierarchie;
- **serverConfig**
  - Server Konfiguration MBeans Hierarchie
- **domainRuntime**
  - Domain Laufzeit MBeans Hierachie
- **serverRuntime**
  - Server Laufzeit MBeans Hierachie
- **edit**
  - Beschreiben einer Domain Konfiguration; EditMBeanServer
- **jndi**
  - read-only JNDI Baum zu dem einzelnen server
- **custom**
  - Auflisten von Custom MBeans

Deployment einer Anwendung

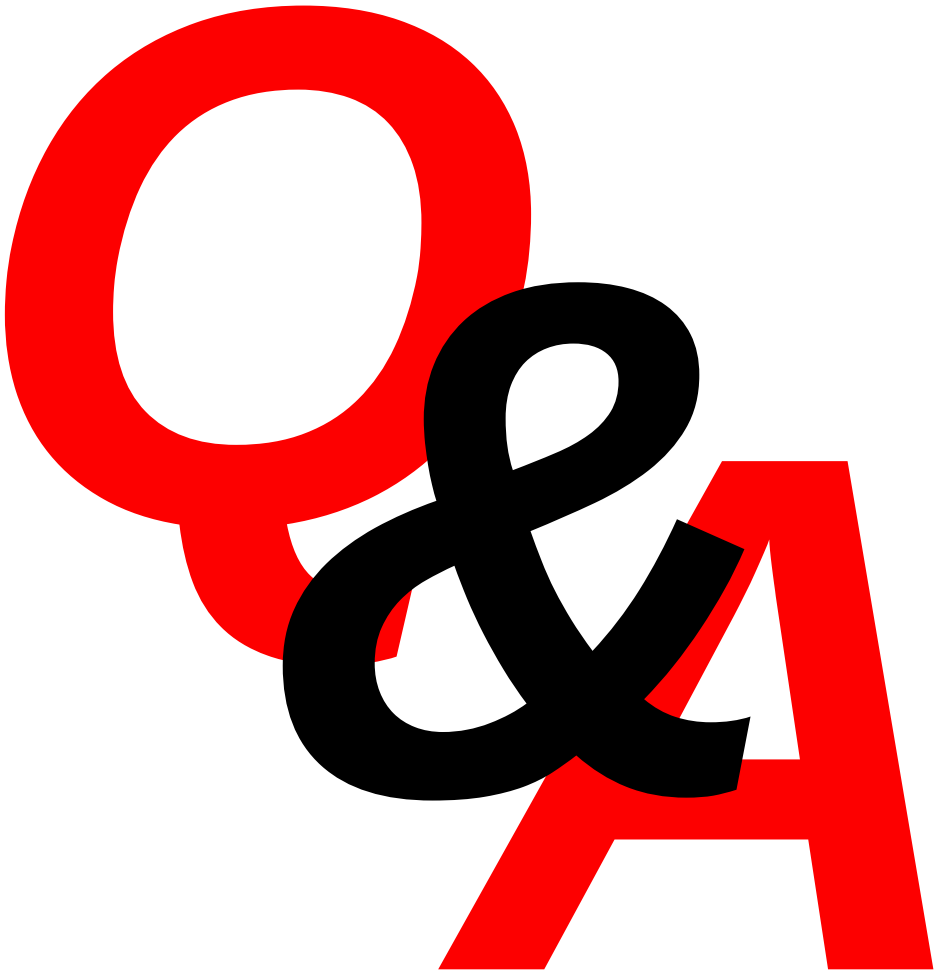
- Im Online Modus
  - Syntax: deploy(appName, path, [targets], [stageMode], [planPath], [options])

```
connect(userConfigFile=UserConfig,userKeyFile=UserKey,url=AdminUrl)
   domainRuntime()
   try:
      print "deploy application: ",appname," from: ",apppath
      progress= deploy(appName=APPNAME, path=APPPATH, targets=TARGET, block='true',
                       clusterDeploymentTimeout=500000, gracefulIgnoreSessions='true',
                       stageMode='nostage')
      startApplication(APPNAME,block='true')
   except:
      print 'Unexpected Exception'

   cd ('/AppRuntimeStateRuntime/AppRuntimeStateRuntime')
   state = cmo.getCurrentState(APPNAME,targets)
   print "State: ", state
```

Reduzieren von WLST Startzeit

- Cache Verzeichnis für Datein:
  - java -Dpython.cachedir="c:\demo\wlst_cache" weblogic.WLST
- Neue Startoption in WLS 10.3: `-skipWLSModuleScanning`
  - Zur Startzeit überprüft WLST die weblogic.jar und alle Klassen, die in der Manifest-Datei referenziert sind und baut den Classpath auf.


Q&A