5

Starting Servers

Objectives

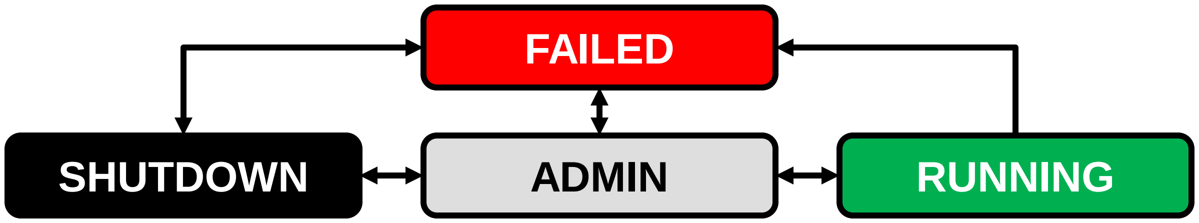
After completing this lesson, you should be able to:

- Start and stop servers with standard scripts
- Deal with server startup problems
- Customize start and stop scripts

WebLogic Server Life Cycle

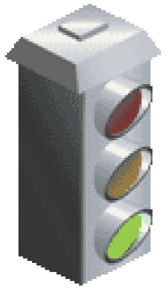
As a server starts, stops, and runs, it finds itself in various states. The most important of these states are:

- RUNNING: The server is fully functional, offering its services to clients.
- SHUTDOWN: The server is configured, but not active.
- ADMIN: The server is running, but available only for administrative operations. Services are available to administrators, but are not available to other clients.
- FAILED: The server itself has failed (for example, it is out of memory) or it has detected that one or more critical subsystems are unstable.



Starting WebLogic Server with a Script

When a domain is created, scripts are generated and placed in the domain's `bin` directory.



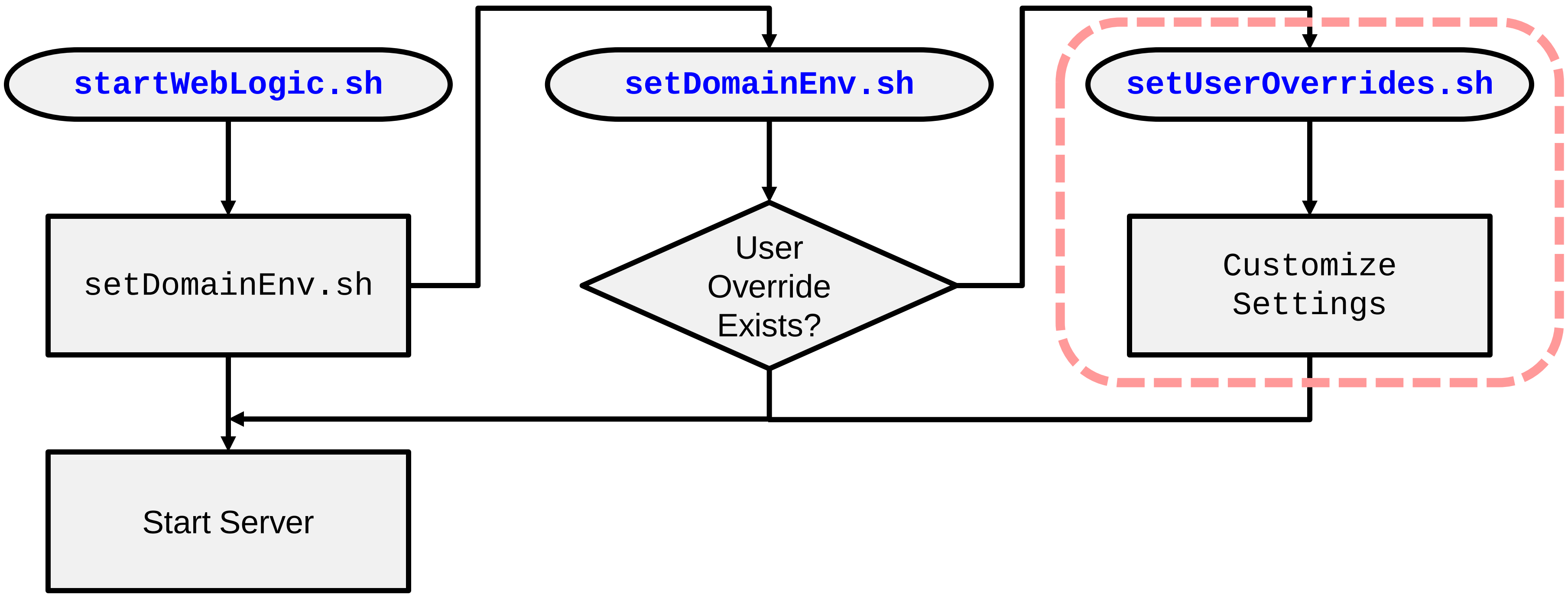
- The administration server start script:
  - `startWeblogic.sh`
    - No parameters

```
$> cd /u01/domains/part1/wlsadmin/bin
$> ./startWebLogic.sh
```

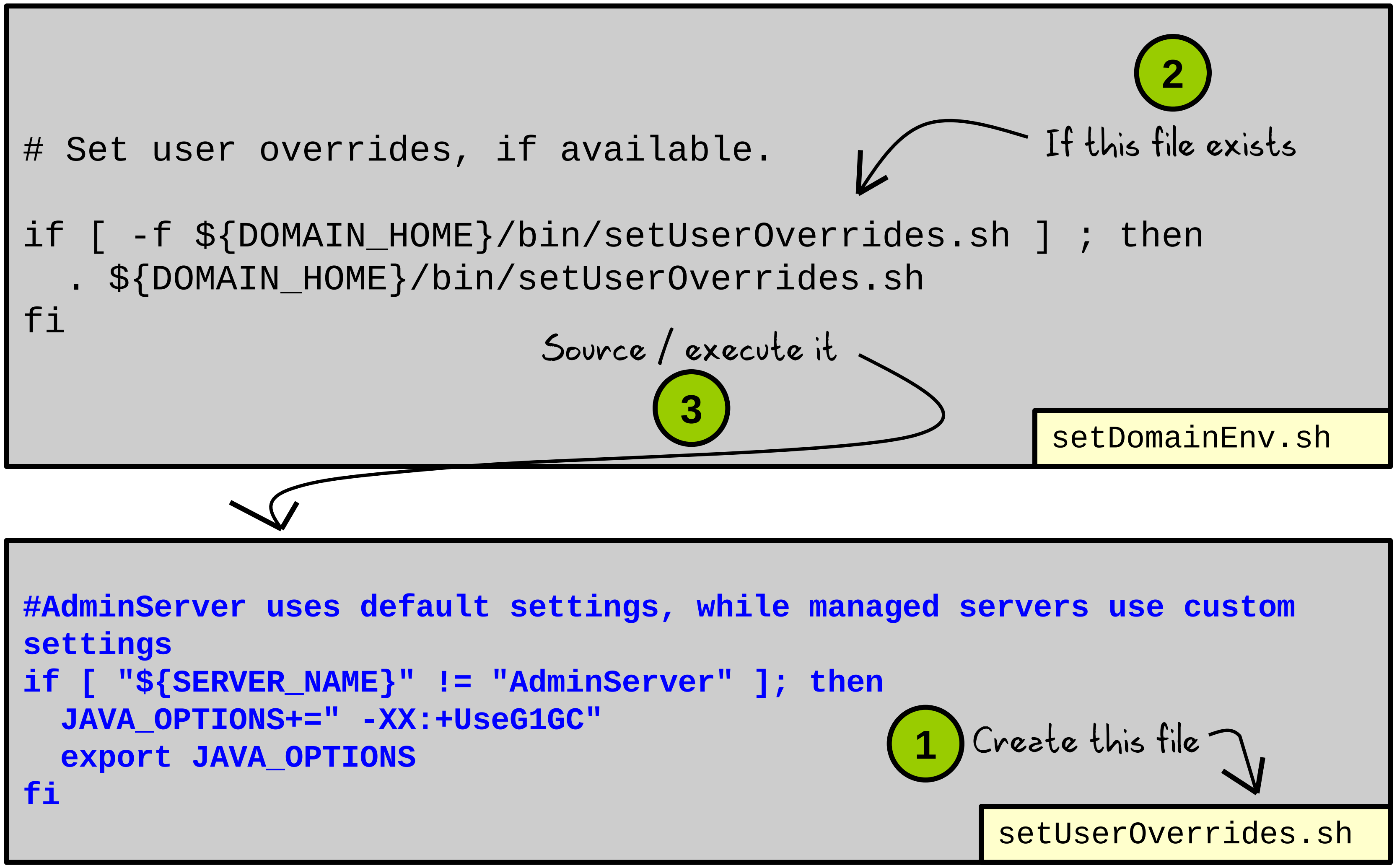
- Managed servers start script:
  - `startManagedWeblogic.sh`
    - Parameter one: The name of the managed server (required)
    - Parameter two: The URL of the admin server (optional)

```
$> ./startManagedWebLogic.sh server1
                        http://host01.example.com:7001
```

Customizing the Scripting Environment



Customizing the Scripting Environment



```
# Set user overrides, if available.

if [ -f ${DOMAIN_HOME}/bin/setUserOverrides.sh ] ; then
  . ${DOMAIN_HOME}/bin/setUserOverrides.sh
fi
```

setDomainEnv.sh

```
#AdminServer uses default settings, while managed servers use custom
settings
if [ "${SERVER_NAME}" != "AdminServer" ]; then
  JAVA_OPTIONS+=" -XX:+UseG1GC"
  export JAVA_OPTIONS
fi
```

setUserOverrides.sh

Creating a Boot Identity File

Starting or stopping WebLogic Server requires administrative authority. The scripts prompt for a username and password.

A boot identity file contains administrator credentials, making prompting unnecessary.

1. Each server has its own directory in the domain. For example, `servers/server1`. Under that directory create a new subdirectory called `security`.
2. Create a text file there called `boot.properties`:

```
username=adminuser
password=thepassword
```

3. Start the server with the script. The server reads the file for its credentials. If they are valid, the server starts. The server also encrypts both values in the file so that they are now secure.

Stopping WebLogic Server

A server can be stopped:

- By using the admin console. There are two options:
  - **When work completes** (sessions complete or time out)
    - You can set the **Graceful Shutdown Timeout** for a server (seconds to wait before switching to a forced shutdown)
  - **Force Shutdown Now** (session data lost)
- With a standard stop script
  - These standard domain scripts do a forced shutdown
- By killing its JVM process
  - `kill -2`
    - Does a forced shutdown
  - `kill -9`
    - "Hard" kill, no "shutdown" code; use only as a last resort

The same effect as pressing **Ctrl+C** in the Terminal window in which a server is running.

Suspend and Resume

A server that is suspended goes from the RUNNING state to the ADMIN state.

- A server can be suspended by using the admin console. There are two options:
  - **When work completes**
  - **Force Suspend Now**

A suspended server can be resumed. It then transitions from the ADMIN (or STANDBY) state to the RUNNING state.

- A server can be resumed by using the admin console.
- STANDBY is a state in which a server has completed its startup procedure, but does not process any requests. (Its regular listen port is closed.) This is used to keep a server ready as a “hot” backup.

Customizing Standard Scripts

The standard start scripts can be customized to:

- Change WebLogic Server runtime options
- Change which JVM runs WebLogic Server
- Change JVM options
  - Performance tuning of the JVM
- Modify the `CLASSPATH`

WebLogic Server Options

The options of `weblogic.Server` are used to:

The Java class that starts WLS

- Change the location of configuration data
  - Examples:
    - WebLogic home: `-Dweblogic.home=`*`path`*
      - Default: Determined by `CLASSPATH`
    - Server root directory: `-Dweblogic.RootDirectory=`*`path`*
      - Default: The directory from which the server was started
- Override a server's configuration
  - Examples:
    - `-Dweblogic.ListenAddress=`*`host`*
    - `-Dweblogic.ListenPort=`*`portnumber`*

This line could be added to a start script such as `startWebLogic.sh`.

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dweblogic.ListenPort=9001"
```

  - Instead, make changes by using the admin console/WLST.

WebLogic Server Options

The options of `weblogic.Server` are used to:

- Change the server type
  - Normally all server services are started.
  - If you do not need EJBs, JCA (Java EE Connector Architecture), or JMS, you can run the lighter version of WebLogic Server that does not start those services.
  - Use the `-DserverType=wlx` option.

```
JAVA_OPTIONS="${JAVA_OPTIONS} -DserverType=wlx"
```

Changing the JVM

- There are environment variables used to indicate which JVM runs WebLogic Server. In `setDomainEnv.sh`, there are the following variables:
  - `VM_TYPE`: Values such as `"HotSpot"` or `"JRockit"`
  - `JAVA_VENDOR`: Values such as `"Oracle"` or `"Sun"`
  - `JAVA_HOME`: Location of the JDK
- If, for example, both the HotSpot and JRockit JDKs are installed, ensure that the `SUN_JAVA_HOME` and `BEA_JAVA_HOME` variables are set properly. Then, near the top of `setDomainEnv.sh`, add a line setting `VM_TYPE`:

```
VM_TYPE="JRockit"
```

or

```
VM_TYPE="HotSpot"
```

  - Logic in the script will set the other variables accordingly.

JVM Options

- The options that are available depend upon the JVM.
- For memory options, set the `USER_MEM_ARGS` variable in `setDomainEnv.sh`.
  - `Xms` is the minimum heap size.
  - `Xmx` is the maximum heap size.
  - `XX:MaxPermSize` sets the maximum permanent space for a HotSpot JVM.

Where Java classes are kept

```
USER_MEM_ARGS="-Xms64m -Xmx200m -XX:MaxPermSize=350m"
```

- For other options, update the `JAVA_OPTIONS` variable in `setDomainEnv.sh`.
  - For example, to choose the “concurrent mark-sweep” garbage collection algorithm of Hotspot:

```
JAVA_OPTIONS="${JAVA_OPTIONS} -XX:+UseConMarkSweepGC"
```

Modifying the `CLASSPATH`

- The Java `CLASSPATH` tells the JVM where to find user-created packages and classes.
- There are many environment variables relating to the `CLASSPATH`. They are used to update the `CLASSPATH` for patches and libraries. These variables are set in:
  - *`domain_name`*`/bin/startWebLogic.sh`
  - *`domain_name`*`/bin/setDomainEnv.sh`
  - `<MW_HOME>/oracle_common/common/bin/commEnv.sh`
- To modify the `CLASSPATH` for:
  - All domains, edit the `commEnv.sh` script
  - A particular domain, edit that domain's `setDomainEnv.sh` script

These files may need to be modified on multiple hosts.

Modifying the CLASSPATH

To modify the `commEnv.sh` installation script:

- Add your JAR file to the `WEBLOGIC_CLASSPATH` environment variable.
  - If your classes need to be found first, place your JAR file as the first thing in the string.

```
WEBLOGIC_CLASSPATH="/tools/mytool.jar${CLASSPATHSEP}..."
```

To modify the `setDomainEnv.sh` domain script:

- Create the PRE_CLASSPATH environment variable, if it does not exist, and assign it to your JAR file.

```
PRE_CLASSPATH="/tools/mytool.jar"
```

- If the variable already exists, add your JAR file to it.

Modifying the `CLASSPATH`

- To add code in a JAR file to the `CLASSPATH` of all servers in the domain, without changing any of the startup or set environment scripts, place the JAR file in the `lib` directory of the domain.
  - All JAR files in this directory are added dynamically to the end of each domain server's `CLASSPATH` the next time the server starts.
  - Note that the JAR file needs to be placed in the domain's `lib` directory on each host on which servers from this domain run.

WebLogic Server Startup Issues

- Severe errors during server startup may cause:
  - WebLogic Server to exit prematurely
  - The JVM to crash (rare)
- Common causes include:
  - Missing or invalid server start parameters
  - Another process using the address and port of the server
  - Custom start scripts with errors
  - Missing or invalid boot credentials or SSL files
  - An invalid or corrupt domain configuration

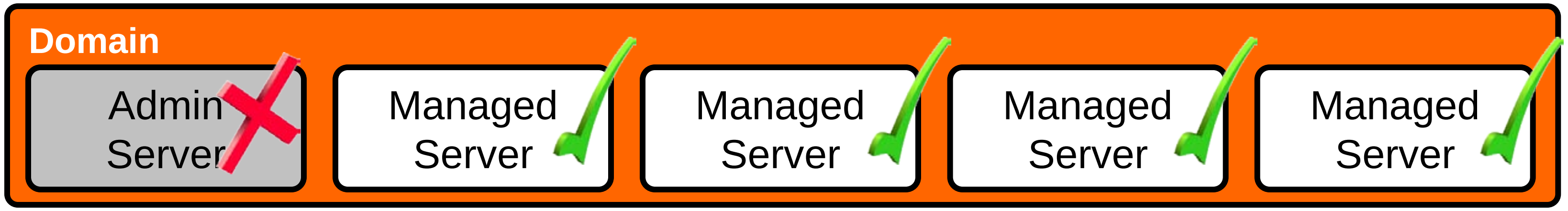
```
<Critical> <WebLogicServer> <BEA-000362> <Server failed.
    Reason: [Management:141245]Schema validation error...>
...
<Error> <WebLogicServer> <BEA-000383> <A critical  service
    failed. The server will shut itself down.>
```

Failed Admin Server

- If the administration server fails:
  - There is no effect on running managed servers or their applications, so your users are unaffected.
    - When the admin server returns, managed servers reconnect.
  - A managed server not running can be started without the admin server, if it has:
    - Been started at least once before to obtain its own copy of the configuration
    - Managed server independence (MSI) mode enabled

Enabled by default

  - No changes to the configuration can be made.



Restarting a Failed Admin Server: Same Machine

- To restart the admin server on the same machine:
  - Run the start script again
    - If the server fails again:
      - Analyze the reason for the failure (from the server log)
      - Correct the issue
      - Run the start script
  - If the admin server was started by Node Manager, Node Manager can be set to automatically restart it.
- When the admin server has restarted, the running managed servers will reconnect to it.

Restarting a Failed Admin Server: Different Machine

- To restart the admin server on a different machine:
  - The admin server backup machine must have already been configured and contain:
    - WLS installation
    - Domain files
    - Deployed applications
  - If using a virtual IP address, move it to the backup machine.
  - Start the admin server (start script or Node Manager).
- When the admin server has restarted, the running managed servers reconnect to it if the admin server has:
  - A virtual IP address
  - A host name that maps to multiple IP addresses

Restarting a Failed Managed Server: Same Machine

- To restart a managed server on the same machine:
  - Run the start script again
    - If the server fails again:
      - Analyze the reason for the failure (from the server log)
      - Correct the issue
      - Run the start script
  - If the managed server was started by Node Manager, Node Manager can be set to automatically restart it.
- As the managed server is restarted, it reconnects with its admin server.

Restarting a Failed Managed Server: Different Machine

- To restart a managed server on a different machine, it needs:
  - WLS installation
  - Domain files
  - Deployed applications
  - JTA and JMS artifacts
  - If using a virtual IP address, move it to the backup machine
- Start the managed server (start script or Node Manager).
- As the managed server is restarted (even if on a different IP address), it reconnects with its admin server.
- A clustered managed server that fails can be configured to migrate to another machine automatically or manually.
  - JTA and JMS artifacts must be available.

Quiz

The `startManagedWebLogic.sh` script takes two parameters. They are:

a. The name of the admin server and the URL of the managed server
b. The name of the managed server and the URL of the managed server
c. The name of the admin server and the URL of the admin server
d. The name of the managed server and the URL of the admin server

Quiz

The name of the boot identity file is:

a. `boot.identity`

b. `identity.properties`

c. `boot.properties`

d. `password.properties`

Summary

In this lesson, you should have learned how to:

- Start and stop servers with standard scripts
- Deal with server startup problems
- Customize start and stop scripts

Practice 5-1 Overview: Starting and Stopping Servers

This practice covers the following topics:

- Starting the administration server with a script
- Starting managed servers with a script
- Creating a boot identity file
- Modifying server start scripts