Intro to Red Hat JBoss Enterprise Application Platform 7

RED HAT DEVELOPER PROGRAM

Table of Contents

Introduction

Red Hat JBoss Enterprise Application Platform 7 (JBoss EAP) is a Java EE 7-certified application platform. It is built on a collection of open source technologies, including an embeddable web server, messaging, clustering and high availability, and caching.

Although JBoss EAP implements the full Java EE 7 specification, it has a structure which allows it to be flexible and as lightweight as the development environment requires. Classes are loaded as modules within the server runtime and are only loaded as needed, which makes for very fast startup times. These classes relate to the subsystems within the application platform, the major functional areas such as logging or app deployments. JBoss EAP also uses the concept of profiles. Profiles define the subsystems, classes, and APIs required for a given runtime; while there is a full Java EE 7 profile defined, the default configuration also includes a lighter web profile, a high availability profile, and a messaging profile which include a subset of the potential subsystems. It is possible to create custom profiles, using a different subset of classes or custom classes. This can reduce the footprint of the server in resource-constrained environments like cloud or container based infrastructures.

JBoss EAP 7 can structure its resources in a more traditional, monolithic server or in a flexible domain hierarchy. A domain divides the application server into two conceptual halves: server configuration and runtime operations. The domain is comprised of four types of systems:

- The domain controller, which defines the profile and subsystem configurations, JVM settings, interfaces, and other settings in a single place.
- Managed servers, which can be installed across multiple systems. These take their configuration from the domain controller and serve the web applications deployed through the domain.
- Hosts, which are where the managed servers are located.
- Server groups, which are configuration nodes. A domain can have multiple profiles, JVM definitions, and web applications. Specific configuration and content is assigned to a server group, and applied to the managed servers within it.

A standalone server has a single, unnamed profile. A managed domain can define many profiles for use by the servers in the domain.

Using the command-line tool

Interactive and noninteractive modes

The JBoss CLI opens up a terminal to allow you to interactively enter commands for the application server. It is also possible to pass commands and have them run immediately, either using a file or in the command line. For example:

```
$ ./bin/jboss-cli.sh --connect --commands=ls\ deployment
```

```
$ ./bin/jboss-cli.sh --connect --file=/usr/jsmith/deployment.txt
```

Command syntax for standalone v managed domain servers

All of the resources within the JBoss EAP installation are structured and accessed in the CLI very much like a directory structure. The subsystems and servers are directories and subdirectories within the overall installation.

When accessing a resource or configuration, it is possible to open that directory and then issue commands. For example:

```
cd subsystem=web
./connector=http:read-resource
```

Or you can run the command with the full path to the resource. For example:

```
/subsystem=web/connector=http:read-resource
```

For a standalone server, all that is necessary is to access the resource or subsystem. However, for a domain, there is a hierarchy of the host, server groups, and managed servers, and the way that the command is structured identifies which of the resources are being accessed, either by the physical configuration (host / server) or the domain configuration (profile / server group).

- Physical configuration
  - Specifying the host only applies to the host and all managed servers on it
  - Specifying the host and server applies changes only to that server
- Profile configuration
  - Specifying the profile applies to the entire domain
  - Specifying the server group applies only to servers within the group

For example, this checks the pool statistics for a datasource in a standalone server:

```
/subsystem=datasources/data-source=ExampleDS/
statistics=pool:read-resource(include-runtime=true)
```

This checks the pool statistics for a datasource for the whole domain:

```
/profile=production/subsystem=datasources/data-source=ExampleDS/
statistics=pool:read-resource(include-runtime=true)
```

This checks the pool statistics for a datasource on a server group:

```
/profile=production/server-group=main-group/subsystem=datasources/
data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
```

This restarts the host and all managed servers on the host:

```
/host=master:start
```

This restarts a single server instance:

```
/host=master/server-config=server-one:start
```

Within a domain configuration (depending on the configuration area), the profile or host must be specified.

NOTE: Most of the examples here are for a standalone server, unless noted otherwise.

Basic operation (command) syntax

The basic structure of an operation is:

```
/resourceType=name:operation(parameter=value)
```

The resource type is the domain location (in a domain) and the subsystem. If the resource is not explicitly given, then the operation runs on the current resource. For example, the first command opens the logging subsystem resource, and the next returns the resource details.

```
cd subsystem=logging
:read-resource(recursive=”true”)
```

To run a command against a child resource, as with a bash shell, use a period and slash (./):

```
cd subsystem=logging
./root-logger=ROOT:change-root-log-level(level=WARN)
```

To run a command against the root resource, use a slash (/):

```
/:read-resource
```

Batch processing

The batch commands allow a group of commands to be executed atomically and to roll back any successful commands if any command in the batch fails. Only operation commands can be in the batch, not shell-like commands such as help, ls, or cd.

Enable batch mode by running the batch command, then enter all of the commands in the batch. You can run the batch using the run-batch command or save it for later use using the holdback-batch command.

For example:

```
batch
```

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_
NAME  --connection-url=CONNECTION_URL --profile=PROFILE_NAME
```

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-
attribute(name=jta,value=true)
```

```
reload
```

```
run-batch
```

Configuration files and directories

The EAP server instances are configured through XML and properties files which define the subsystems, settings, network configuration, and other definitions for the application server. There are different configuration files which can be loaded for high availability profiles and web or full profiles.

Standalone configuration files

`standalone.xml`	Default configuration.
`standalone-ha.xml`	The default configuration, plus mod_cluster and JGroups for high availability.
`standalone-full.xml`	The default configuration, plus messaging and IIOP.
`standalone-full-ha.xml`	The default configuration, plus all of the subsystems for clustering, high availability, and messaging.

Domain configuration files

`domain.xml`	Defines all of the profiles (high availability, full, default) and subsystems for the entire domain. This is only read by the domain master.
`host.xml`	Identifies an individual managed server within the domain.

Top-level directories

`bin`	Startup scripts, startup configuration files, and various command line utilities like vault, add-user, and Java diagnostic report available for Unix and Windows environments.
`domain`	Configuration files, deployment content, and writable areas used by the managed domain processes run from this installation.
`modules`	JBoss EAP is based on a modular class loading architecture. The various modules used in the server are stored here.
`standalone`	Configuration files, deployment content, and writable areas used by the single standalone server run from this installation.

Directories with standalone or domain

`configuration`	Configuration files for the server.
`data`	Persistent data storage.
`log`	Log directory.
`tmp and tmp/auth`	Temporary directory used by the server. tmp/auth is for authentication tokens for local clients.
`servers/ServerName`	Domain only. Instance directory for managed servers within a domain. Each application server instance will have its own subdirectory, with instance-specific subdirectories: • data • log • tmp
`lib/ext`	Standalone only. For installed JARs referenced by applications using Extension-List.
`deployments`	Standalone only. Directory to drop content that can be automatically detected and loaded into the runtime.

Basic management

`./bin/jboss-cli.sh --connect`	Connect to the CLI.
`./bin/standalone.sh`	Start a standalone EAP server.
`./bin/domain.sh`	Start an EAP domain controller.
`Ctrl + C in the terminal`	Stop an EAP standalone or domain controller interactively.
`shutdown`	Stop an EAP standalone or domain controller running in the background.
`/host=master:start`	Start a server in a domain.
`/host=master:stop`	Stop a server in a domain.
`:suspend(timeout=60)`	Suspend a standalone server.
`:suspend-servers(timeout=60)`	Suspend all servers in a domain.
`/host=master/server-config=server-one:suspend(timeout=60)`	Suspend a single server in a managed domain.
`/server-group=main-server-group:suspend-servers(timeout=60)`	Suspend all servers in a server group.

`:resume`	Resume a server.
`./bin/add-user.sh`	Add a user interactively.
`./bin/add-user.sh -u 'mgmtuser1' -p \\ 'password1!' -g 'guest,mgmtgroup'`	Add a user with parameters for the username (-u), password (-p), and user groups (-g).
`patch apply /path/to/downloaded-patch.zip` `shutdown --restart=true`	Patch a server.
`patch apply /path/to/downloaded-patch.zip --host=Host` `shutdown --restart=true`	Patch a host in a managed domain.
patch rollback --patch-id=PATCH_ID --reset-configuration=TRUE shutdown --restart=true	Roll back a patch.

6 About the author

DEON BALLARD is a middleware marketing writer for Red Hat and writes for and edits the Middleware Blog. She has previously worked as a journalist, designer, and as a technical writer for identity management and security, cloud and virtualization, system management, and middleware software.

@JBoss

http://middlewareblog.redhat.com