Understanding Extensions, Subsystems, and Profiles

Goals

After completing this section, you should be able to do the following:

• Describe the architecture of extensions, subsystems, modules and profiles.

• Describe the relationship between JBoss EAP 8 modules, extensions, and subsystems.

• Describe how JBoss modules affect application class loading.

Modules in EAP

EAP 8 is based on a core framework provided by the WildFly Core project that, among other tasks, manages module loading and activation, following the architecture provided by the JBoss Modules project. Basically, a module provides code (Java classes) to be used by EAP services or by applications.

The following figure shows the core components within EAP 8. The upper half shows components that are part of the WildFly Core project (core EAP), mostly modules. The bottom half shows modules provided by other JBoss community projects, which are integrated into the WildFly Full project, and then customized and supported by Red Hat as JBoss EAP 8.

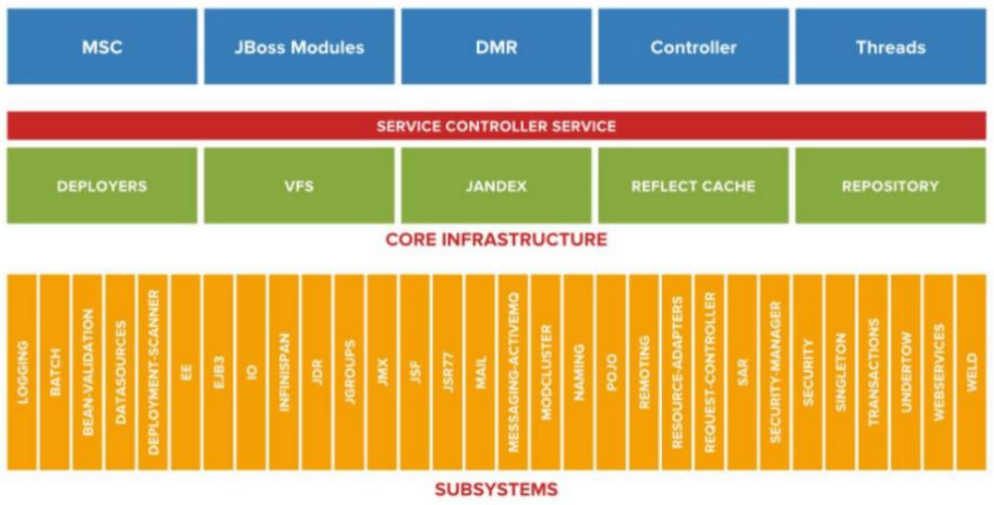

Figure 1.16:
EAP 8 Advanced Architecture

Modules are loaded in an isolated ClassLoader, and classes from other modules can only be seen when implicitly requested. This means that a module can be implemented without any concern regarding possible conflicts with the implementation of other modules. All code that runs in EAP (including code provided by the core) runs inside modules. Application code is no exception, so applications are isolated from each other as well as from EAP services.

The JBoss Modules architecture allows fine-grained control of code visibility. One application may see a module that exposes a particular version of an API, while another application may see a second module that exposes a different version of the same API.

An application developer can control this visibility manually and it can be very useful in some scenarios. But for the most common cases, EAP 8 automatically decides which modules to present to an application, based on its use of the JEE APIs.

All modules available in an EAP 8 installation are folders within JBOSS_HOME/modules:

- The JBOSS_HOME/modules/system/layers/base folder contains the modules provided by the EAP 8 product.
- Third-party products that add modules to EAP 8 are expected to create their own modules in JBOSS_HOME/modules/system/layers.
- The system administrator is expected to add local modules as folders, directly in JBOSS_HOME/modules.

Within one of the module folders, the name of a module is used to create a folder tree, much like compiled Java classes. For example, the extension named org.wildfly.extension.undertow is located in JBOSS_HOME/modules/system/layers as org/wildfly/extension/undertow.

Modules and extensions

A module that provides functions and capabilities to the application server is called an extension. An extension does not simply provide a code, but also provides a management model, which consists of one or more subsystems, that allow the core to configure the extension.

The EAP core only loads and activates a module (and consequently an extension) when it is required, so functions that are not in use do not waste memory or CPU. For example, a non-clustered EAP server instance has no clustering overhead.

Most of the EAP functionality that is related to the deployment and configuration of Jakarta EE applications and servers is implemented through extensions.
A subset of all extensions provided by EAP is sufficient to implement the Jakarta EE web profile. Add a few more extensions, and EAP will meet the full Jakarta EE profile. But if the Jakarta EE web profile is too much for an application, you can use a smaller subset of the available extensions.

EAP 8 even provides some extensions that add capabilities beyond the Jakarta EE 10 standards. One example of this is clustered singletons.

Extensions are added to an EAP instance by using the <extension> element, which appears at the start of the main EAP configuration file (standalone.xml or domain.xml). The following XML list shows some of the extensions set in the standalone.xml configuration file out of the box:

```
<?xml version="1.0" ?>
<server xmlns="urn:jboss:domain:4.1">
    <extensions>
        <extension module="org.jboss.as.clustering.infinispan"/>
        <extension    module="org.jboss.as.clustering.jgroups"/>
        <extension              module="org.jboss.as.connector"/>
        <extension     module="org.jboss.as.ee"/>     <extension
        module="org.jboss.as.ejb3"/>
... remainder of configuration file ... </server>
```

Adding an extension to the configuration files does not fully load that plugin or make it active. It just makes the extension manageable, which means that the extension's management model is loaded into memory and can be modified using EAP 8 management tools.

For example, cluster configurations can be done if the related extensions are available, but the related modules will only be loaded and activated when an application that
requires cluster services is deployed. If a clustered application is not deployed, none of the clustering-related extensions will be activated, even if the clustering extensions management model was modified by the administrator.

use

If an organization's applications do not need a particular extension, the system administrator can disable the extension so that EAP no longer loads its management model. Simply remove the <extension> entry and its corresponding <subsystem> section (discussed below) from the configuration file. You can do this by editing the XML file directly or by using the management CLI.

Extensions and subsystems

An extension management model provides one or more subsystems, which can be configured and managed using the management API provided by the core, through the management console, and the management CLI. When an extension is added to an EAP instance, the capabilities and attributes of the subsystems provided by the extension's management model are configured within the <subsystem> element in the EAP configuration file.

Most of the time, an extension module provides a single subsystem, as illustrated in the following figure:

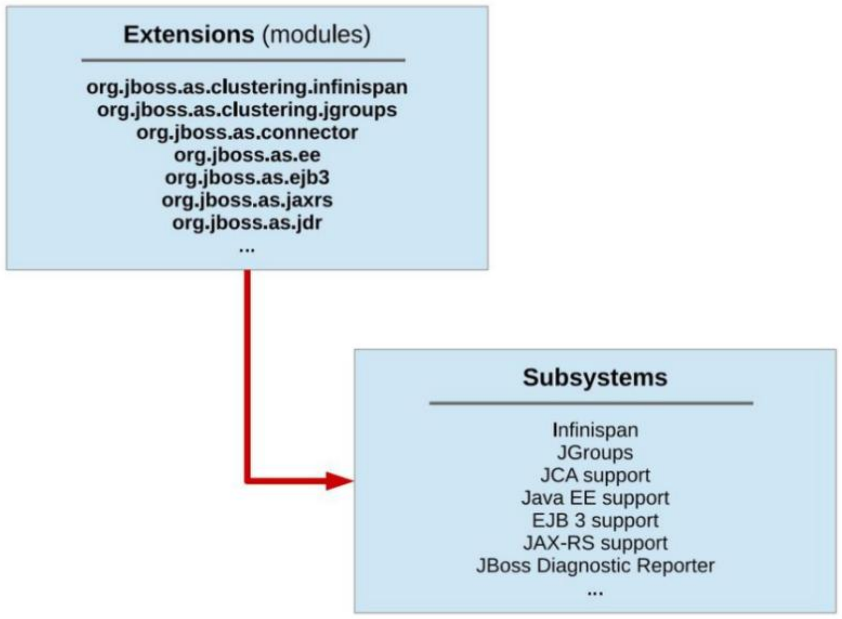

Figure 1.17:
EAP 8 Extensions and Subsystems

For example, the org.jboss.as.jpa extension provides the jpa subsystem. The configuration of this subsystem, in the factory standalone.xml configuration file, looks like this:

```
<subsystem xmlns="urn:jboss:domain:jpa:1.1"> <jpa default-
    datasource="" default-extended-persistence-inheritance="DEEP"/> </subsystem>
```

The most frequently used subsystems can be configured using the simple forms and wizards provided by the administration console. All subsystems can be configured using the CLI, so there is usually no need to edit the XML directly to configure your subsystems.

Items that appear within a subsystem are completely unique to that particular subsystem. In fact, each subsystem has its own XML schema to define what is allowed within the <subsystem>
element. All EAP 8 subsystem definitions can be found in the JBOSS_HOME/docs/schema folder.

use

If a subsystem does not require any specific value, an empty <subsystem> entry is still required in the configuration file. For example, the jaxrs subsystem is configured at the factory with no particular value.

```
<subsystem xmlns="urn:jboss:domain:jaxrs:1.0"/>
```

Subsystems and profiles

A subsystem configuration is not maintained by itself in EAP 8 configuration files. A set of subsystem configurations is grouped within a <profile> element.

The standalone server instance's standalone.xml configuration file contains a single, anonymous profile definition. The managed domain domain.xml configuration file contains four factory-predefined profiles that should meet most use cases for applications deployed on EAP:

1. **default: These are the most frequently used subsystems, which includes logging, security, datasources, infinispan, weld, webservices , ejb3. The default profile implements not only the Jakarta EE web profile, but also most of the full Jakarta EE profile.**

2. **ha (high availability): contains exactly the same subsystems as the profile default, plus clustering capabilities, provided primarily by the jgroups subsystem.**

3. **full (complete): It is similar to the default profile, but mainly adds the messaging (messaging-activemq) and other less used subsystems.**

4. **full-ha (full high availability): same as full profile, but adds capabilities of clusters.**

The following figure illustrates the four factory profiles in the factory domain.xml configuration file:

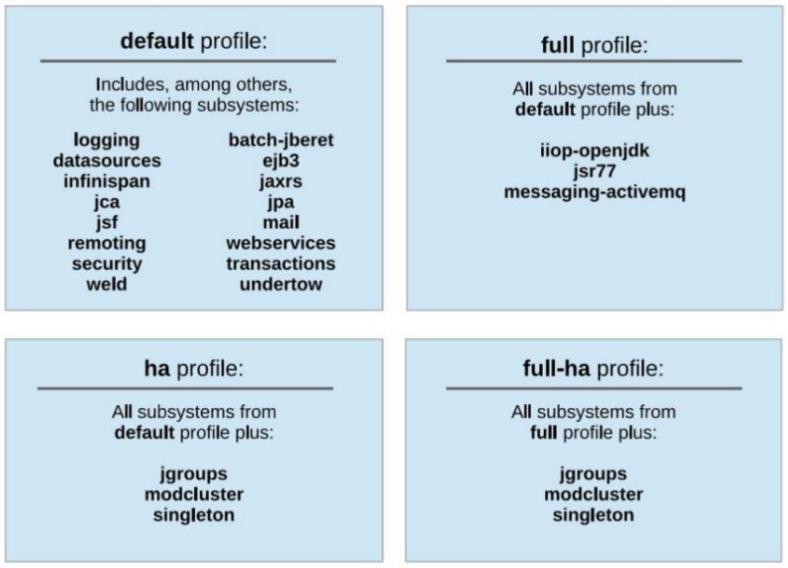

Figure 1.18:
EAP 8 profiles

Out-of-the-box profiles can be used by different server instances in the same managed domain, but the profiles are not directly associated with server instances. The server instances are pooled. The profile is associated with the group.

EAP profiles must be customized to a particular application and environment needs, so two actual instances of the standalone server will likely have different profiles, with different subsystem configurations.

use

See the JBOSS_HOME/standalone/configuration folder. Notice that there are four separate configuration files:

- standalone.xml: which can be compared with the default profile in domain.xml.
- standalone-ha.xml – which can be compared to the ha profile in domain.xml.
- standalone-full.xml: which can be compared with the full profile in domain.xml.
- standalone-full-ha.xml: which can be compared to the full-ha profile in domain.xml.

They are provided so that a standalone server instance can be started with more or fewer subsystems available.

In a managed domain, the administrator can create new profiles, either from scratch or cloned from the factory-provided ones, and then customize them before associating them with their respective groups. A profile can also be a child of another profile, thus inheriting its parent profile's subsystem settings, so that common settings can be shared and maintained in one place.

On a standalone server instance, the unique anonymous profile must be customized. But later this course will show how to start a standalone server instance using a configuration file with a name other than standalone.xml, so that the original file is kept for reference.

use

Be careful not to confuse an EAP profile with a Jakarta EE profile. An EAP profile is a group of subsystem configurations used to define the capabilities and services of an EAP server instance. A Jakarta EE profile is a set of Jakarta EE standards, that is, JSR.

Questionnaire: Selection of profiles

Using the profiling information provided on the previous pages, choose the correct answer to the following questions:

1. Which profiles use the messaging-activemq subsystem? (Choose two options.)

a. **default**
b. **ha**
c. **full**
d. **full-ha**

2. What profiles use the registration subsystem? (Choose only one option).

a. **default**
b. **ha**
c. **full**
d. **full-ha**
It is. **All of them**

3. Which profile would you use if you had a Jakarta EE application that used message-driven EJBs, ie the messaging- activemq subsystem, and you were NOT planning to set up a cluster of server instances? (Choose only one option).

a. **default**
b. **ha**
c. **full full-**
d. **ha**
It is. **None of them You would need to create a custom profile.**

4. Which profile would you use for a WAR file deployed to a group of four clustered server instances? Assume that this WAR contains a simple web application that requires only database access. (Choose only one option).

a. **default**
b. **ha**
c. **full**
d. **full-ha**
It is. **None of them You would need to create a custom profile.**

5. Which profile would you use if you needed to deploy a clustered application that requires IIOP support for remote EJBs, ie the openjdk-iiop subsystem? (Choose only one option).

a. default

b. ha

c. full

d. full-ha

It is. None of them. You would need to create a custom profile.

6. Which profile would you use for a WAR file deployed to a single server? Assume that this WAR contains a simple web application that requires only database access. (Choose only one option).

a. default

b. ha

c. full

d. full-ha

It is. None of them You would need to create a custom profile.

Solution

Using the profiling information provided on the previous pages, choose the correct answer to the following questions:

1. Which profiles use the messaging-activemq subsystem? (Choose two options.)

- **a.** default
- **b.** ha
- **c.** **full**
- **d.** **full-ha**

2. What profiles use the registration subsystem? (Choose only one option).

- **a.** default
- **b.** ha
- **c.** full
- **d.** full-ha
- **It is.** **All of them**

3. Which profile would you use if you had a Jakarta EE application that used message-driven EJBs, ie the messaging-activemq subsystem, and you were NOT planning to set up a cluster of server instances? (Choose only one option).

- **a.** defaul
- **b.** t ha
- **c.** **full**
- **d.** full-ha
- **It is.** None of them You would need to create a custom profile.

4. Which profile would you use for a WAR file deployed to a group of four clustered server instances? Assume that this WAR contains a simple web application that requires only database access. (Choose only one option).

- **a.** default
- **b.** **ha**
- **c.** full
- **d.** full-ha
- **It is.** None of them You would need to create a custom profile.

5. Which profile would you use if you needed to deploy a clustered application that requires IIOP support for remote EJBs, ie the openjdk-iiop subsystem? (Choose only one option).

a. default

b. ha

c. full

d. **full-ha**

It is. None of them You would need to create a custom profile.

6. Which profile would you use for a WAR file deployed to a single server? Assume that this WAR contains a simple web application that requires only database access. (Choose only one option).

a. **default**

b. ha

c. full

d. full-ha

It is. None of them You would need to create a custom profile.