redhat TRAINING

CHAPTER 6

CONFIGURATION OF SOURCES DATA

General description	
Meta	Configure JDBC database drivers and data sources. • Describe the
Goals	purpose and architecture of the subsystem. of data sources. • Implement a JDBC driver as a module and configure the driver in the data source subsystem. • Configure a data source. • Configure an XA data source. •
sections	Exploration of the data sources subsystem (and questionnaire) • Configuring JDBC drivers (and guided exercise) • Configuration of data sources (and guided exercise) • Configuration of an XA data source (and exercise guided)
Laboratory work	• Configuration of data sources

Exploring the data sources subsystem

Goals

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

- **Describe the database subsystem and connection pools.**
- **Describe how to validate connections to the database.**

data sources

Most Java EE applications comprise a database that contains information that users need to access and perform basic create, read, update, and delete operations. A data source is a facility provided by an application server responsible for accessing databases without providing sensitive information, such as credentials and the location of the database, to an application. To connect to a database, the application uses a name bound to the data source through the application server. For a JEE application server, the name is defined by the Java Naming and Directory Interface (JNDI) specification.

Using a Java Database Connectivity (JDBC) API, specific to a user's relational database management system, users can tune a data source to maximize efficiency and manage load on the database. data.

Data sources are configured through the *data source subsystem.* In order to establish a new database, two separate steps must be followed:

1. **Install the JDBC driver for a particular database server.**
2. **Define a data source within the datasource subsystem of the output file. setting.**

Database connection pools

One hurdle to overcome when managing a data source is being able to handle traffic and concurrent connections to the database. It can cause a bottleneck and performance issue during peak loads of deployed applications.

In order to improve database connection performance, EAP creates a *database connection pool* for each database connection used by applications. Pooling contains open connections to the database so that when the application needs to access the database, an open connection is already available, thus avoiding the overhead of opening and closing a connection for each individual request . The connection pool size can be configured for each data source.

use

Each database used by an application requires its own connection pool.

Database connection validation EAP 7 can handle power outages, network problems, database maintenance, or any other event that may cause the server to lose connection to the database. Using one of several validation methods, users can configure and customize when and how to validate the database connection and how to handle connection loss. There are two main methods for validating connections:

- **Validate on match**

  Every time a connection is removed from the connection pool, it will be validated. This method offers the fastest connection recovery, but also creates the greatest load on the database.

  If the connection is found to be invalid, the next connection in the pool is used and a warning is logged. The server continues to cycle through the connection pool until a valid connection is found. If no connection in the pool is valid, a new one is created. If the connection creation also fails, an exception is thrown.

- **Background validation**

  The connection is validated based on a customizable time value. By setting the validation time to a lower value, the connection is checked more frequently and can remove more invalid connections from the pool, but it also requires more resources and a higher load on the database.

Questionnaire: Data sources

Based on the previous class, choose the correct answer to the following questions:

1. Which default validation method represents the highest stress in the database load? (Choose one option).

a. **Validate on match**

b. **background validation**

2. What listed items are required to create a data source? (Choose two options).

a. **Install the relevant JDBC driver.**

b. **Configure connection validation.**

c. **Define the data source in the EAP configuration file.**

d. **Adjust the size of the connection pool.**

3. What term describes a facility provided by a Java EE application server to open multiple database connections? (Choose one option).

a. **Data source**

b. JDBC

c. **Database connection pooling JNDI**

d.

4. Which Java API is responsible for defining names in a data source? (Choose One option).

a. JDBC b.

JNDI

c. **JBDS**

d. JPA

5. Which statements describe advantages of using a connection pool? (Choose three options).

a. **It provides a simple naming convention for accessing the database.**

b. **Connection pools improve performance by keeping a pool of available connections, thus avoiding creating a new physical connection each time it is needed.**

c. **The connection pool size is customizable to maximize database efficiency for each data source.**

d. **Checking database connections can be accomplished in the background or before each new connection.**

It is. **Connection pools protect secure connection credentials.**

Solution

Based on the previous class, choose the correct answer to the following questions:

1. Which default validation method represents the highest stress in the database load? (Choose one option).

a. **Validate on match**

b. background validation

2. What listed items are required to create a data source? (Choose two options).

a. **Install the relevant JDBC driver.**

b. Configure connection validation.

c. **Define the data source in the EAP configuration file.**

d. Adjust the size of the connection pool.

3. What term describes a facility provided by a Java EE application server to open multiple database connections? (Choose one option).

a. Data source

b. JDBC

c. **Database connection pooling** JNDI

d.

4. Which Java API is responsible for defining names in a data source? (Choose One option).

a. JDBC b.

JNDI

c. JBDS

d. JPA

5. Which statements describe advantages of using a connection pool? (Choose three options).

a. It provides a simple naming convention for accessing the database.

b. **Connection pools improve performance by keeping a pool of available connections, thus avoiding creating a new physical connection each time it is needed.**

c. **The connection pool size is customizable to maximize database efficiency for each data source.**

d. **Checking database connections can be accomplished in the background or before each new connection.**

It is. Connection pools protect secure connection credentials.

Configuring the JDBC drivers

Goals

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

- **Implement a JDBC driver as a module.**
- **Configure a controller in the data source subsystem.**

Implementation of a JDBC controller as a module

A JDBC driver is a component that Java applications use to communicate with a database. A JDBC driver is packaged in a JAR file (Java file), which contains a class file with the definition of the driver. JDBC drivers are available from database providers, such as MySQL or PostgreSQL.
To use a JDBC driver in EAP 7, it must first be installed as a module on the server.

A JDBC driver module can be added manually or much more simply with the EAP CLI. The CLI add module command requires the user to provide the following information: • Name – A name based on the name of the Java
package that EAP will use to store the JAR file and a configuration file named module. It must be unique and cannot conflict with existing libraries available in the $JBOSS_HOME/ modules directory.

- **Resources – Refers to the location that stores the JAR file.**
- **Dependencies – Identify which Java libraries are used by the JDBC driver and where they are located in EAP.**

The syntax of the module add command is:

```
[disconnected /] module add --name=<module_name> --resources=<JDBC_Driver> --
dependencies=<library1>,< library2>,...
```

For example, the following command in the EAP CLI will create the MySQL 5.5 JDBC module using the mysql-connector-java.jar file downloaded from the database vendor:

```
[disconnected /] module add --name=com.mysql \ --
resources=/opt/jboss-eap-7.0/bin/mysql-connector-java.jar \ --
dependencies=javax.api,javax.transaction.api
```

use

Because this command does not modify the standalone.xml or domain.xml files, the EAP server does not need to be running and the CLI does not need to be in connected mode.

After running this command, a new directory is created at /opt/jboss-eap-7.0/modules/com/mysql/main. The new directory contains the provided JAR file, as well as module.xml, which is generated based on the dependencies given by the driver.

The following module.xml file defines a module for the MySQL JDBC driver:

```
<?xml version="1.0" ?>

<module xmlns="urn:jboss:module:1.1" name="com.mysql">

    <resources>
        <resource-root path="mysql-connector-java.jar"/>
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/> </
    dependencies> </
module>
```

use

To remove an EAP module, the server must be stopped so that the following command can be used in the CLI:

```
[disconnected /] module remove --name=<module_name>
```

Definition of controllers

After adding the driver as a module, the next step is to create a <driver> definition in the <drivers> section of the data source subsystem in the EAP configuration file.

A CLI operation can be used to easily create it, without touching the XML file. The following command can be run on a standalone server:

```
/subsystem=datasources/jdbc-driver=<driver_name>:add\ (driver-module-
 name=<module_name>,driver-name=<unique_driver_name>)
```

In the define driver command, the following fields are required: • driver-name – A unique name for the driver.

- **driver-module-name: the unique name of the module installed in the directory $JBOSS_HOME/modules.**

For managed domain mode, the syntax is as follows:

```
/profile=<profile_name>/subsystem=datasources/jdbc-driver=< driver_name>:add\
(driver-module-
name=<module_name>,driver-name=<unique_driver_name>)
```

For example, the CLI command to set the MySQL driver to the default profile in a managed domain looks like this:

```
[domain@172.25.250.254 /] /profile=default/subsystem=datasources/jdbc-driver=mysql:add(\
                    driver-module-name=com.mysql,\
                    driver-name=mysql\
)
```

use

The domain.xml/standalone.xml configuration file will have an additional driver tag, similar to the following after defining the driver using the CLI:

```
<drivers>
       <driver name="mysql" module="com.mysql"/>
</drivers>
```

After the <driver> is defined in the <drivers> section of the data source subsystem, the driver is referenced via the name attribute. After completing this step, users can create data sources that use the driver.