Data source configuration

Goals

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

- **Configure a data source.**
- **Configure connection validity for data sources.**

Creating a data source In EAP 7, a data

source is configured in the server configuration file (domain.xml or standalone.xml) within the datasource subsystem.

The structure of the data source subsystem is as follows:

```
<subsystem xmlns="urn:jboss:domain:datasources:4.0"> <datasources>

    <datasource jndi-name="..." jta="true or false" pool-name="..." enabled="true
or false" use-java-context="true or false" >
        <connection-url> connection URL </connection-url>
        <driver> the driver module </driver>

        <pool>
            Connection pool settings </
        pool>

        <security>
            Username and password for connecting to the database </security>

        <validation>validation settings</validation>
        <timeout>timeouts</timeout>

        <statement>SQL statements</statement> </
    datasource>
    <drivers>

        <driver name="..." module="module_name"/>
    </drivers> </
datasources> </
subsystem>
```

1. **The JNDI name used to look up the data source.**
2. **Indicates if JTA integration is enabled or disabled.**
3. **The name of the pool referenced in the <pool> child element.**
4. **Indicates whether the data source is enabled or disabled.**
5. **Indicates whether to bind the data source to the global JNDI.**
6. **The JDBC driver connection URL.**
7. **The connection set configuration.**
8. **Security credentials to connect to the database.**
9. **Validation strategies for data source connections.**
10. **Contains child elements that are timeout values.**

11 Describes configuring the behavior of prepared statements.
12 The definition of the controller.

Any number of <datasource> entries can be defined within the <datasources> subsystem. Each data source requires a unique JNDI name that is assigned through the jndi-name attribute. Components that need to obtain a connection to a database from a pool use the jndi-name value to look up the pool in the JNDI name service. A valid JNDI name for a data source must begin with "java:" or with "java:/jboss", for example, java:/jboss/ datasources/bookstore.

Various configuration settings for connection pooling are included in the <pool> section. Some of the pool values are <min-pool-size>, <max-pool size>, and <prefill>; the latter attempts to complete the connection pool when it is initially created.

Various values are included in the <validation> section to verify the validity of a connection to a database in the pool. Connection validation will be discussed later in this section.

The *{2}<statement>{2}* section is intended for configuring the behavior of prepared statements. The available options are <prepared-statement-cache-size> to specify the number of prepared statements for caching per connection, and <share-prepared-statements> to specify whether prepared statements can be reused without closing.

use

Another <statement> setting is <track-statements>, which can be set to true, false, or nowarn. This is a useful option because, when set to true or nowarn, it closes connections to databases and result sets that were not explicitly closed by the application when the connection returns to the pool.

The <drivers> section allows you to define JDBC drivers implemented as modules. The name attribute for the controller corresponds to the value used in the <datasource> definition. The module attribute must be assigned to a module deployed on the server.

use

The XSD schema file for the Data Source Subsystem section is located in the docs/schema folder of the EAP installation, in a file named wildfly-datasources_4_0.xsd.

Database connection validation EAP 7 provides the ability to manage 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.

The first step to enable validation is to select one of the following validation methods and set the value to true within the <datasource> section of the server configuration file in the <validate> section: •
validate-on-match

If the validate-on-match value is set to true, each time a connection is removed from the connection pool, it will be validated.

Here is an example of using validate-on-match:

```
<datasources>
   <datasource ...>
      <validation>
         <validate-on-match>true or false</validate-on-match>  (1)
         <exception-sorter class
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>  (2)
      </validation>
   </datasource>
</datasources>
```

1. **Can be true or false to enable or disable the match validation option.**
2. **The class name for the exception classifier, specific to the database provider.**

• background-validation

If the value of background-validation is set to true, the connection will be validated based on the value set in <background-validation-millis>.

Here is an example of using background-validation:

```
<datasources>
   <datasource ...>
      <validation>
         <background-validation>true or false</background-validation> <exception-  (1)
         sorter class
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>  (2)
      </validation>
   </datasource>
</datasources>
```

1. **Can be true or false to turn context validation on or off.**
2. **The class name for the exception classifier, specific to the database provider.**

use

Both validation methods can be defined, although only one can be enabled at a time.

After selecting the validation method, users must select a validation mechanism to describe how the connection will be validated. Select one of the following mechanisms to validate the connection: • valid-connection-checker

EAP 7 provides several classes that can be used to validate a user relational database management system-specific database connection. For example, for a MySQL validation class checker, the following appears:

```
<valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/
>
```

• check-valid-connection-sql

This validation mechanism requires the user to provide a simple SQL statement, which will be executed to validate the connection. Below is an example of a MySQL database connection:

```
<check-valid-connection-sql>select 1</check-valid-connection-sql>
```

• exception-sorter

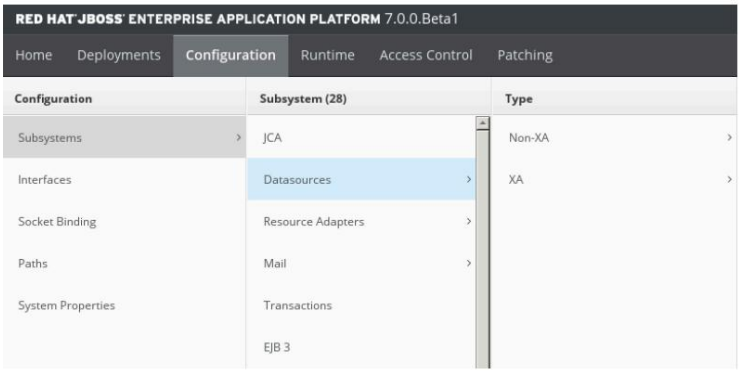
Using exception-sorter allows users to provide a class to properly catch and clean up after fatal connection exceptions. EAP 7 offers several exception classifier classes depending on the relational database management system that is being used. Here is an example of a MySQL exception classifier:

```
<exception-sorter class
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
```

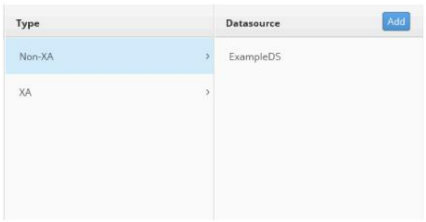
Create a data source with the admin console

The EAP administration console offers an easy way to create a data source using preconfigured templates for different database providers. The following steps can be used to create a non-XA data source with the admin console on a standalone server.

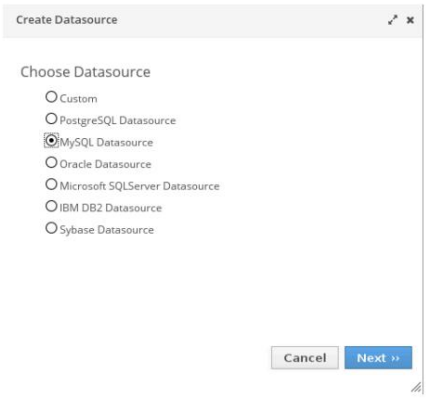
1. Select Settings at the top of the management console, select Subsystems and then Data Sources to access the data source subsystem.



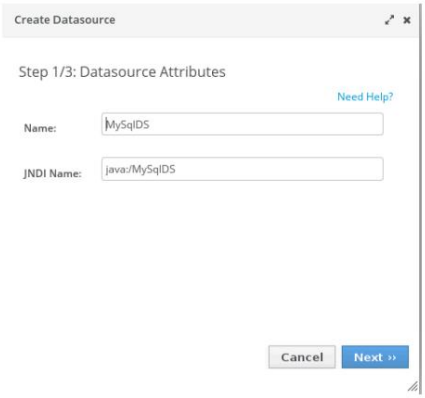
2. Click Non-XA and then click Add to open the Create Font wizard of data.



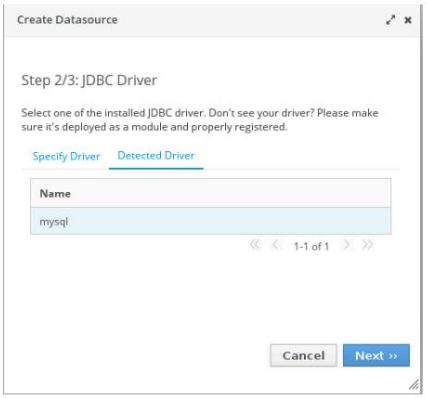
3. Select the database to be used by the Java application and click Next. If neither is valid, select the Custom option and refer to the JDBC driver documentation to complete the following steps.



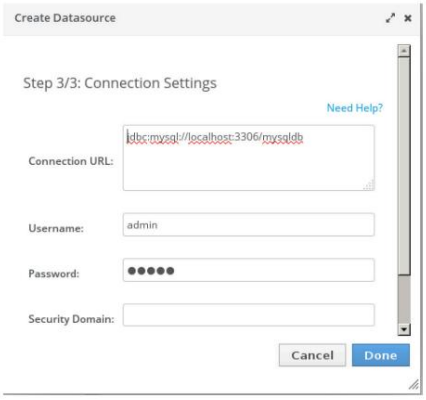
4. After entering the Data Source Attributes, click Next to select controller.



5. Click the Detected Driver option and select the mysql driver to be detected. installed as a module and select Next.



6. In the final step, notice that the connection URL is formatted for the correct MySQL syntax. Click Done to finish creating the data source.



Connection pool test

The administration console has a Test Connection button to verify that the connections in a connection pool can access the database. To access, you must click on the name of the data source and on the Test connection button.

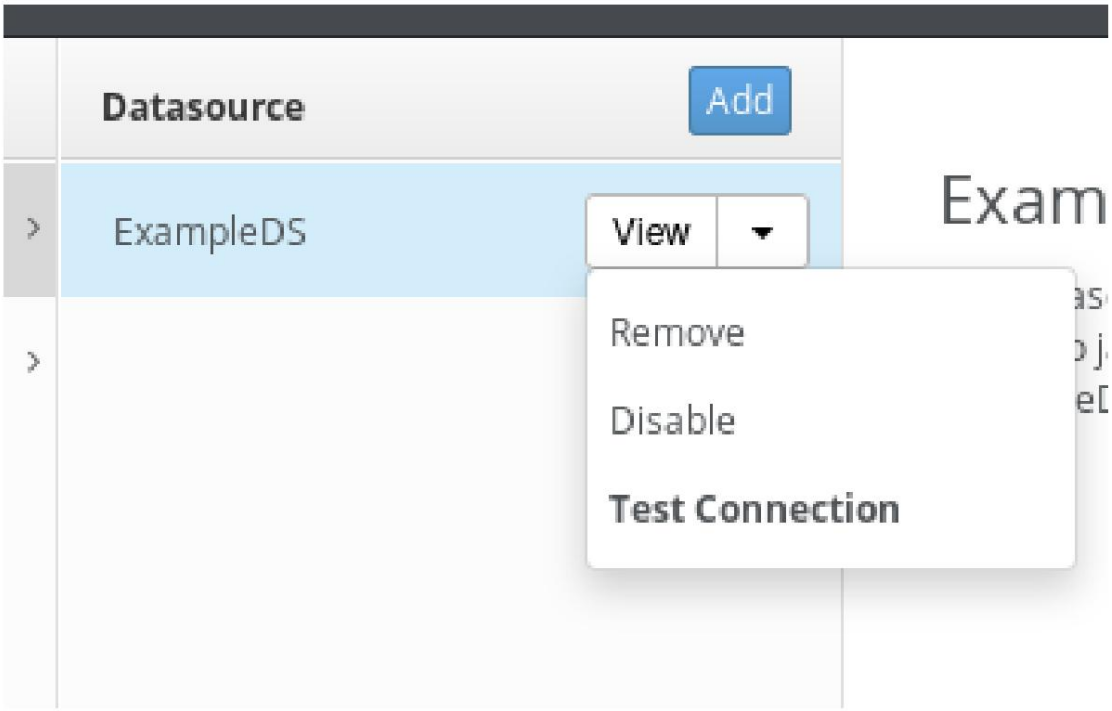


Figure 6.7: Test connection button

In the same way, the CLI can be used to test whether the data source was configured correctly. To do this, use the following command:

```
[standalone@localhost /] /subsystem=datasources/data-source=datasource_name:test  connection-in-
pool
```

Data Source Example

The following data source configuration is for a MySQL database named MySqlDS.

```
<datasources>
    <datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
        <connection-url>jdbc:mysql://localhost:3306/jbossdb</connection-url>
        <driver>mysql</driver>
        <security>
            <user-name>admin</user-name>
            <password>admin</password> </
        security>
        <validation>
            <valid-connection-checker class
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
            <validate-on-match>true</validate-on-match> <background-
            validation>false</background-validation> <exception-sorter class

name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
        </validation>
    </datasource>
</datasources>
```

1. The JNDI name to use to look up the data source per component on the EAP server.
2. The name refers to the driver defined in the <drivers> section below the data source.
3. The credentials used to access the MySQL database.
4. Both background and match-on-match validation are defined, although only one can be enabled. In this case, the data source will validate each connection as it is removed from the connection pool using the org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker class.

The corresponding CLI command to define this data source is as follows:

```
[standalone@localhost] data-source add --name=MySqlDS --jndi-name=java:jboss/MySqlDS \ --driver-
name=mysql \ --connection-
url=jdbc:mysql://localhost:3306/jbossdb \ --user-name=admin --
password=admin \ --validate-on-match=true --
background-validation=false \ --valid-connection-checker-class-name=\

org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker \ --exception-
sorter-class-name=\
org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

A similar command can be used while in domain mode in the CLI to add a data source to a specific profile:

```
[domain@localhost] data-source add --name=MySqlDS --profile=full-ha \ --jndi  name=java:jboss/
MySqlDS --driver-name=mysql \ --connection-
url=jdbc:mysql://localhost:3306/jbossdb \ --user-name=admin --
password=admin \ --validate-on-match=true --
background-validation=false \
--valid-connection-checker-class-name=\
org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker \ --exception-
sorter-class-name=\
 org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```