Module 3

Exercise: *Defining a Jdbc log handler*

Exercise objective

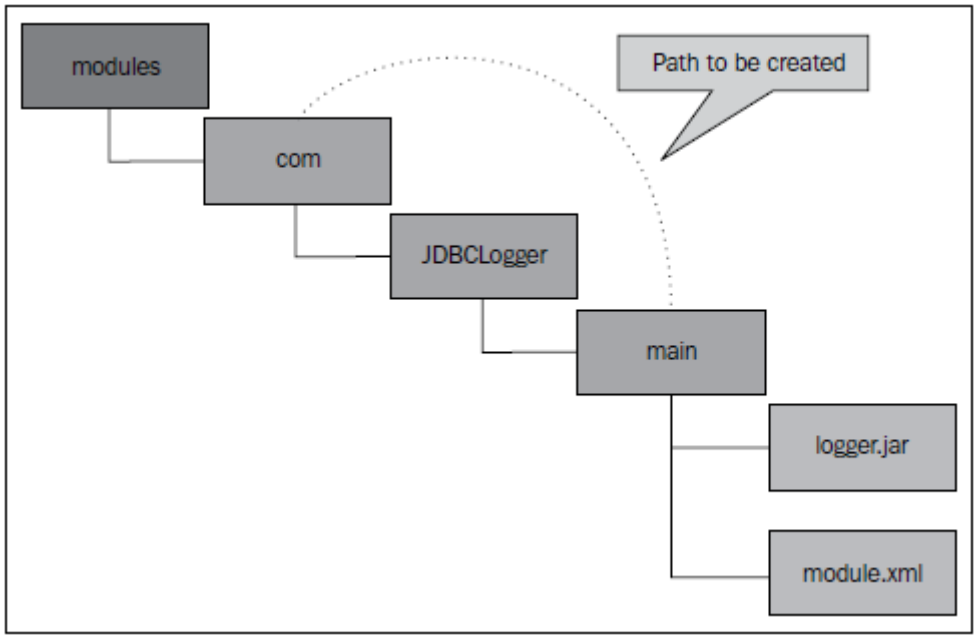
Add a custom log handler to handlers list.

Preparation

In order to add a custom handler, you need to define class that extends the **`java.util.logging.Handler`** abstract class and overrides its abstract methods. You are given a JDBC log handler called **`com.example.loger.JdbcLogger`** which is used to write the logs on database. Read the class and try to understand what it does.

Tasks

`com.example.loger.JdbcLogger` is packaged in an archive file called **`logger.jar`**. Install it as a module into the application server. Name the module as **`com.JDBCLogger`**, which requires the following structure under the modules folder:

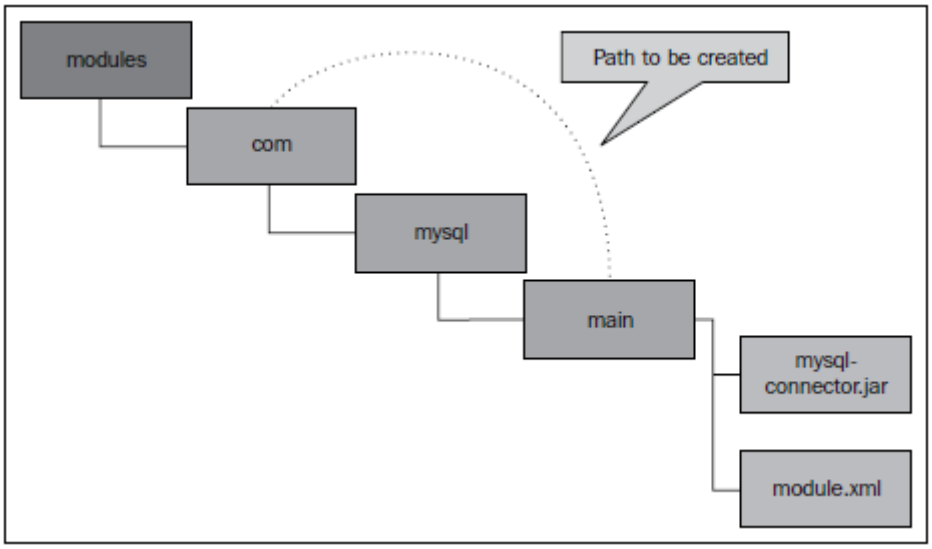


The label **Path to be created** shows off the directory structure under which we will place the **logger.jar** archive and its configuration file (**module.xml**), which follows here:

```
<module xmlns="urn:jboss:module:1.0" name="com.JDBCLogger">
    <resources>
        <resource-root path="logger.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="org.jboss.logging"/>
        <module name="com.mysql"/>
```

</dependencies>
</module>

Please notice that this module has a dependency on another module named **com.mysql**. The JDBC driver used in this example, also known as Connector/J can freely be downloaded from the MySQL site: (**http://dev.mysql.com/downloads/connector/j**).



The **module.xml** contains the actual module definition. The module name is **com.mysql**.Next, you need to state the path to the JDBC driver resource and finally the module dependencies.

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
   <resources>
     <resource-root path="mysql-connector-java-5.1.17-bin.jar"/>
   </resources>
   <dependencies>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
   </dependencies>
</module>
```

We are almost done. Now, insert the handler in the logging subsystem, which contains within its properties the database connection strings and the statement that will be used to insert logs into the database:

```
<custom-handler name="DB" class="com.example.logger.JdbcLogger"
 module="com.JDBCLogger">
   <level name="INFO"/>
   <formatter>
   <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t)
   %s%E%n"/>
   </formatter>
   <properties>
     <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
     <property name="jdbcUrl" value=
                      "jdbc:mysql://localhost:3306/mylog"/>
     <property name="username" value="root"/>
     <property name="password" value="root"/>
```

```
      <property name="insertStatement" value="insert into log_table
      values (?, $TIMESTAMP, $LEVEL, $MDC[ip], $MDC[user], $MESSAGE,
      hardcoded)"/>
      </properties>
   </custom-handler>
   <root-logger>
      <level name="INFO"/>
      <handlers>
            <handler name="CONSOLE"/>
            <handler name="FILE"/>
            <handler name="DB"/>
      </handlers>
    </root-logger>
```

The new handler, named DB, is enlisted in the root-logger to collect all logging statements that have a priority of INFO or above. Before testing the logger, don't forget to create the required tables on your MySql database:

```
CREATE TABLE log_table(
      id INT(11) NOT NULL AUTO_INCREMENT,
      `timestamp` VARCHAR(255) DEFAULT NULL,
      level VARCHAR(255) DEFAULT NULL,
      mdc_ip VARCHAR(255) DEFAULT NULL,
      mdc_user VARCHAR(255) DEFAULT NULL,
      message VARCHAR(1500) DEFAULT NULL,
      hardcoded VARCHAR(255) DEFAULT NULL,
      PRIMARY KEY (id)
) ENGINE = INNODB AUTO_INCREMENT = 1 ;
```