**create Datasource using com.oracle.database.jdbc.ojdbc**

create Datasource using com.oracle.database.jdbc.ojdbc



+2

sources

To create a DataSource using the Oracle JDBC driver (com.oracle.database.jdbc.ojdbc), follow these steps:

**Steps to Create a DataSource**

**1. Include the Required Oracle JDBC Driver**

Ensure that the Oracle JDBC driver (ojdbc) JAR file is included in your project. For example:

* Use ojdbc8.jar for Java 8 or later.
* Add the dependency in your build tool, such as Maven:

xml

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc8</artifactId>

<version>21.4.0.0.1</version>

</dependency>

Alternatively, download the JAR file from Oracle's website or Maven repositories and add it to your classpath[5](https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc8/21.4.0.0.1/usages).

**2. Configure Database Properties**

Create a db.properties file to store database connection details. Example:

text

ORACLE\_DB\_DRIVER\_CLASS=oracle.jdbc.driver.OracleDriver

ORACLE\_DB\_URL=jdbc:oracle:thin:@localhost:1521:orcl

ORACLE\_DB\_USERNAME=your\_username

ORACLE\_DB\_PASSWORD=your\_password

**3. Implement the DataSource Factory**

Use the OracleDataSource class provided by the Oracle JDBC driver to configure and create a DataSource.

Example Java implementation:

java

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.sql.SQLException;

**import** java.util.Properties;

**import** javax.sql.DataSource;

**import** oracle.jdbc.pool.OracleDataSource;

**public** **class** OracleDataSourceFactory {

**public** **static** DataSource getOracleDataSource() {

Properties props = **new** Properties();

FileInputStream fis = **null**;

OracleDataSource oracleDS = **null**;

**try** {

*// Load properties from file*

fis = **new** FileInputStream("db.properties");

props.load(fis);

*// Configure OracleDataSource*

oracleDS = **new** OracleDataSource();

oracleDS.setURL(props.getProperty("ORACLE\_DB\_URL"));

oracleDS.setUser(props.getProperty("ORACLE\_DB\_USERNAME"));

oracleDS.setPassword(props.getProperty("ORACLE\_DB\_PASSWORD"));

} **catch** (IOException | SQLException e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (fis != **null**) fis.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**return** oracleDS;

}

}

This factory class reads database properties from a file and initializes an OracleDataSource instance[2](https://www.digitalocean.com/community/tutorials/java-datasource-jdbc-datasource-example).

**4. Test the DataSource**

Write a test program to verify the connection:

java

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** javax.sql.DataSource;

**public** **class** DataSourceTest {

**public** **static** **void** main(String[] args) {

DataSource ds = OracleDataSourceFactory.getOracleDataSource();

**try** (Connection con = ds.getConnection();

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT empid, name FROM Employee")) {

**while** (rs.next()) {

System.out.println("Employee ID=" + rs.getInt("empid") + ", Name=" + rs.getString("name"));

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

This program retrieves employee data from the database using the configured DataSource[2](https://www.digitalocean.com/community/tutorials/java-datasource-jdbc-datasource-example).

**5. Optional: Use Connection Pooling**

For better performance, use connection pooling libraries like Apache Commons DBCP or configure pooling directly in your application server.

Example with Apache Commons DBCP:

java

**import** org.apache.commons.dbcp.BasicDataSource;

**public** **class** DBCPDataSourceFactory {

**public** **static** DataSource getPooledDataSource() {

BasicDataSource ds = **new** BasicDataSource();

ds.setDriverClassName("oracle.jdbc.driver.OracleDriver");

ds.setUrl("jdbc:oracle:thin:@localhost:1521:orcl");

ds.setUsername("your\_username");

ds.setPassword("your\_password");

ds.setInitialSize(10); *// Initial connections in pool*

ds.setMaxActive(20); *// Maximum connections in pool*

**return** ds;

}

}

This approach provides efficient connection management for high-performance applications[2](https://www.digitalocean.com/community/tutorials/java-datasource-jdbc-datasource-example).

**6. Application Server Configuration**

If deploying on an application server, configure the DataSource via administrative tools or XML configuration files as per server guidelines (e.g., WebSphere, Tomcat)[1](https://www.ibm.com/docs/en/was/8.5.5?topic=adaa-creating-data-source-that-uses-oracle-jdbc-driver-oci)[2](https://www.digitalocean.com/community/tutorials/java-datasource-jdbc-datasource-example).

By following these steps, you can successfully create and test a DataSource using com.oracle.database.jdbc.ojdbc.