/ JAVA DB SCHOOL
Maven & JDBC

/ Maven

What is Maven?

- Project management tool for Java
  - Used for projects build, dependency and documentation
  - Simplifies the build process

- Usages:
  - Build projects – JAR, WAR etc
  - Add jars and dependencies
  - Provide project information
  - Update dependencies

Core concepts (1)

- POM (Project Object Model):
  - XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project
- Dependencies and repositories:
  - Dependencies are external Java libraries
  - Repositories are directories of packaged JAR files
  - The local repository is a directory on the machine hard drive
  - If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in the local repository

Core concepts (2)

- Build Life Cycles, Phases and Goals:
  - A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals
  - A Maven command is the name of a build lifecycle, phase or goal
  - If a lifecycle is requested, all build phases in that life cycle are executed
  - If a build phase is requested, all build phases before it in the defined sequence are executed

Core concepts (3)

- Build Profiles:
  - Configuration values which allows to build the project using different configurations
- Build Plugins:
  - Are used to perform specific goals

Build lifecycle

- 3 built-in lifecycles:
  - Default – handles project deployment
  - Clean – handles project cleaning
  - Site – handles documentation

Default build phases

- Validate - validate the project is correct and all necessary information is available
- Compile - compile the source code of the project
- Test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- Package - take the compiled code and package it in its distributable format
- Verify - run any checks on results of integration tests to ensure quality criteria are met
- Install - install the package into the local repository, for use as a dependency in other projects locally
- Deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects

Minimal POM

- Project root
- modelVersion - should be set to 4.0.0
- groupId - the id of the project's group.
- artifactId - the id of the artifact (project)
- version - the version of the artifact under the specified group

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

Reference: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

Pros and cons

- Pros:
  - Add all the dependencies required for the project automatically by reading pom file
  - Easy to build the project to jar, war etc.
  - Easy to start the project in different environments
  - Easy to add new dependencies
- Cons:
  - Environment setup
  - Cannot add dependencies if the dependency is not available in a repository

/ Maven with IntelliJ

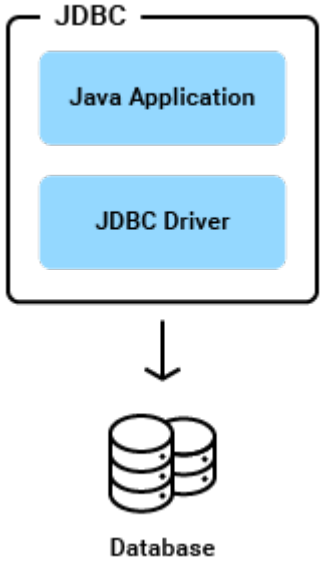
/ JDBC

JDBC - Java Database Connectivity

- Standard abstraction (API) for Java applications to communicate with various databases
- JDBC along with the database driver is capable of accessing databases and spreadsheets

How does it work

- Establishes a connection with a data source
- Sends queries and update statements to the data source
- Processes the results



Reference: https://www.progress.com/faqs/datadirect-jdbc-faqs/how-does-jdbc-work

JDBC API

- DataSource – used to establish connections
- Connection – controls the connection to the database. An application uses the connection object to create statements.
- Statement, PreparedStatement, CallableStatement – used for executing SQL statements
- PreparedStatement – used when an application plans to reuse a statement multiple times
- CallableStatement – used to call stored procedures that return values
- ResultSet – contains the results of a query. It provides methods for iterating through the results of the query

/ Cheat sheet

Cheat sheet (1)

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.17</version>
    </dependency>
</dependencies>
```

Cheat sheet (2)

```
String connectionUrl = "jdbc:mysql://localhost:3306/<database-name>";
String username = "username";
String password = “password";

Connection connection = DriverManager.getConnection(connectionUrl,
username, password);
```

Cheat sheet (3)

```
Statement ps = connection.createStatement();
ResultSet rs = ps.executeQuery(“SELECT * FROM myTable”);

while (rs.next()) {
    MyObject c = new MyObject(rs.getInt(“column1"), rs.getString(“column2"));
}
```

Cheat sheet (4)

```
PreparedStatement ps = connection.prepareStatement(“INSERT INTO
`myTable` (`column1`, `column2`) VALUES ( ?, ?);”);

ps.setInt(1, 123);
ps.setString(2, “myValue”);
ps.execute();
```

/ Practice, practice, practice

Practice

- Implement the following methods for table **customers**
  - getById
  - getAll
  - update
  - insert
  - delete
- Implement methods for:
  - Adding a new order for an existing customer
  - Viewing all orders for an existing customer

/ Homework

Homework

- Implement the remaining methods from the practice section

- Implement the following methods:
  - Update the status of one order (id given as parameter)
  - Add comments to one order (id given as parameter)
  - When placing an order update the stock for the products

/ Q&A


OBILE/ACADEMY