



MySQL Week 7 Exercises

Background

The weekly exercises are designed to augment the video lessons. In the exercises, you will develop a menu-driven application in Java. This application will demonstrate how to perform CRUD (Create, Read, Update, and Delete) operations on a MySQL database.

You will be working in a Project schema (database) that contains do-it-yourself (DIY) projects. A DIY project contains project details, materials, steps, and categories. Below is a diagram of the tables and relationships in the Project schema. Don't worry at this point if you don't understand what the diagram is telling you. This will become clear soon. For now, just know that there are five tables in the Project schema: project, material, step, category, and project_category. This is what you will build in the exercises.



There will be a final project in this (MySQL) part of the back-end course. These exercises will help prepare you for that.



MySQL Week 7 Exercises

Instructions

URL to GitHub Repository: <https://github.com/Frandito-A/Week07SQL.git>

URL to Public Link of your Video: <https://youtu.be/6HEcZbTZBIA>

Instructions:

1. Follow the [Exercises](#) below to complete this assignment.

- In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., mysql-java) and any .sql files that you create.
- Include the functionality into your Video when you see: 📹
- Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project. Don't forget to include the requested functionality, indicated by: 📹
 - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
 - Your video should be a maximum of 5 minutes.
 - Upload your video with a public link.
 - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

2. In addition, please include the following in your Coding Assignment Document:

- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

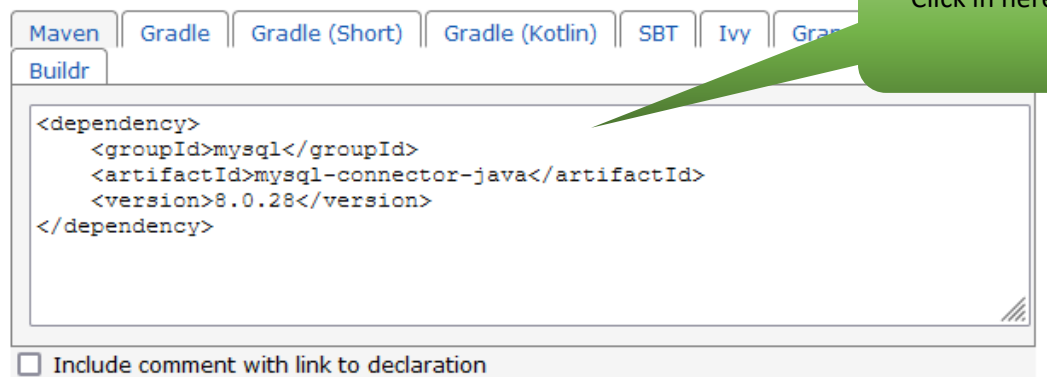
- Push the .pdf to the GitHub repo for this week.
- Upload the .pdf to the LMS in your Coding Assignment Submission.

1. Enter the values in the table below into the fields and click "Finish".



MySQL Week 7 Exercises

1. In this step, you will add the MySQL driver as a dependency in the dependencies section.
 - a. Below the `<properties>` element, create a new element named `<dependencies>` with the closing tag `</dependencies>` below it.
 - b. In a browser, navigate to <https://mvnrepository.com/>. Type "mysql" into the search box and press Enter.
 - c. Find "MySQL Connector/J" (most likely the first result) and click the link "mysql-connector-j". Click on the most recent version number link.
 - d. You will see a page with information about the dependency in a table followed by several tabs and a box with the dependency. Click in the box and the Maven dependency is copied to the clipboard.



- e. Paste the contents of the clipboard into the `<dependencies>` section in pom.xml. The dependencies section should look like this:

```
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
  </dependency>
</dependencies>
```

2. In this step, you will need to add a section to the POM that tells Eclipse which compiler version to use when compiling your project. This is done by adding a definition for the Maven compiler plugin.
 - a. In a browser, navigate to the Maven compiler usage page: <https://maven.apache.org/plugins/maven-compiler-plugin/usage.html>.
 - b. Find the section "Configuring Your Compiler Plugin". Copy the entire `<build>` section to the clipboard.




MySQL Week 7 Exercises

- c. Paste the clipboard contents into pom.xml below the `<dependencies>` section. Replace the XML comment inside the configuration element with a reference to the Java version defined in the properties section. To add a property reference, surround it with `${}`. The section should look like this:


```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>${java.version}</source>
          <target>${java.version}</target>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

At this point, if you are hopelessly lost, refer to the solutions section below.

3. Show all of your pom.xml. 

Set up the project packages

In this section you will create the project structure by adding some packages. This will all be done in the Package Explorer panel in Eclipse.

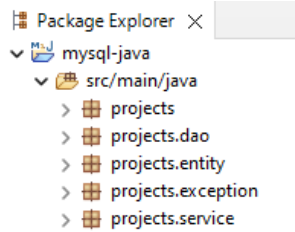
1. Create the following packages and subpackages under `src/main/java`. Show the package structure in Package Explorer. 

- a. `projects`
- b. `projects.dao`
- c. `projects.entity`
- d. `projects.exception`
- e. `projects.service`

Your screen shot should look like this:



MySQL Week 7 Exercises



Create an exception class

In this section you will create an exception class that will be used in the `mysql-java` project. This is an unchecked exception that will be used throughout the application. We do this because all of the exceptions thrown by the Java Database Connectivity (JDBC) API classes are checked `SQLException` objects. In coding the application, you will turn the checked exceptions into unchecked exceptions to keep your code clean.

In this section, you will create the class `DbException` in the `projects.exception` package.

1. In the `projects.exception` package, create a class named "DbException". This class must extend `RuntimeException`. Override the following constructors from the superclass:

```
public DbException(String message) {}  
  
public DbException(Throwable cause) {}  
  
public DbException(String message, Throwable cause) {}
```

Be sure to call the matching constructor in the superclass from each constructor in

`DbException`. Show `DbException` class in your video. 

Create the JDBC connection class

In this section, you will create a class that obtains a `JDBC Connection` object from the driver manager. When you call `DriverManager.getConnection()`, the driver manager looks up the MySQL driver and loads it. It then establishes a TCP connection between the application and a MySQL server. If the connection cannot be made for some reason, the driver manager throws a checked `SQLException`. This is converted to an unchecked exception in a `catch` block.

Follow these steps to create the `JDBC Connection` class.


1. Create a class in the `projects.dao` package named `DbConnection`. In the class, create the following constants: `HOST`, `PASSWORD`, `PORT`, `SCHEMA`, and `USER`. Set the constants to the correct values. If you followed the instructions above and created a user in MySQL Workbench, the constants should look like this:



PROMINEO TECH

MySQL Week 7 Exercises

```
private static String HOST = "localhost";  
private static String PASSWORD = "projects";  
private static int PORT = 3306;  
private static String SCHEMA = "projects";  
private static String USER = "projects";
```


2. In the `DbConnection` class, create a method named `getConnection()`. It should be public and static and should return a `java.sql.Connection` object. In the `getConnection()` method:
 - a. Create a String variable named `uri` that contains the MySQL connection URI.
 - b. Call `DriverManager` to obtain a connection. Pass the connection string (URI) to `DriverManager.getConnection()`.
 - c. Surround the call to `DriverManager.getConnection()` with a try/catch block. The catch block should catch `SQLException`.
 - d. Print a message to the console (`System.out.println`) if the connection is successful.
 - e. Print an error message to the console if the connection fails. Throw a `DbException` if the connection fails.
 - f. Show this entire class. 

Create the main class

Every Java application must have an entry point. This is a class with a `main()` method. In this section, you will create a class with a `main()` method. From the `main()` method you will temporarily call `DbConnection.getConnection()` to test that you can obtain a connection to the MySQL server.

Follow these steps to create the application entry point.

1. Create a class in the `projects` package named `ProjectsApp`. The class must have a `main()` method.
2. In the `main()` method, call the `DbConnection.getConnection()` method.

Run the Java application. Show your console showing that the connection succeeded. 

Final touches

Typically, when you push a project to GitHub, you only want to push source code, not built class files or extra configuration files maintained by a tool such as Eclipse. In a Maven project, all built classes are put in subdirectories off of the `target` directory. So, the `target` directory should not be pushed to GitHub.

Likewise, Eclipse maintains its project configuration in a directory named `".settings"`. This directory should also not be pushed to GitHub.



MySQL Week 7 Exercises

To exclude files or directories from being pushed to GitHub, put the directory and file names into a file named `.gitignore`. This file must be in the project root directory. Simply put the paths of the files or directories on separate lines in `.gitignore` to tell git to ignore these files or directories.

To ignore files, put the file name relative to the project root. Separate directory names with slashes ("/"). To ignore directories, put the directory name on a separate line in `.gitignore` and add a slash ("/") on the end.

Follow these steps to instruct git to ignore the `.settings` directory and the `target` directory.

1. In the root of the Eclipse project, create a file named `".gitignore"`. It should contain the following lines:

```
.settings/
```

```
target/
```