KIV/PIA Labs Example Application Project

Spring implementation of a simple web application. Demonstrates the applications student's should be able to create after passing the course.

1. Get acquainted with contents of the pom.xml file.
2. Build the project using `mvn install` and examine contents of the created archive.
3. Study the following section on handling database connections.
4. Study the section on passing arguments to queries.

Importing Maven Project into Your IDE

While these tutorials may be slightly outdated due to new version releases,
the main course of action usually remains the same.

Import into Eclipse

Import into Netbeans

[Import into Intellij IDEA] (https://www.jetbrains.com/idea/help/importing-project-from-maven-model.html)

Database Connection Management

This section provides overview of possible ways to close database connections maintained by JDBC. Many good reasons
to do so can be found elsewhere.

Pre-JDK 7

In Java 6 and earlier, you have to close your resources manually:

```
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet set = null;

    try {
```

```
        conn = DriverManager.getConnection("some url");
        stmt = conn.prepareStatement("some query");
        set = stmt.executeQuery();

        //process results

    } catch (SQLException e) {
        //handle exception
    } finally {
        if(set != null) {
            try {
                set.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }

        if(stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }

        if(conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }
    }
```

This is however rather verbose, so we can write a utility method:

```
     public void close(Connection conn, PreparedStatement stmt, ResultSet s
et) {
          if(set != null) {
               try {
```

```
                set.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }

        if(stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }

        if(conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                //nothing left to do here, log this event and continue
            }
        }
    }
```

So the actual call looks like this:

```
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet set = null;

    try {

        conn = DriverManager.getConnection("some url");
        stmt = conn.prepareStatement("some query");
        set = stmt.executeQuery();

        //process results

    } catch (SQLException e) {
        //handle exception
    } finally {
```

```
        close(conn, stmt, set);
    }
```

Using Existing Utilities

Reinventing the wheel is not productive, so let's use existing library.

1. Go to the Maven Repository Global Catalogue.
2. Search for **commons dbutils**
3. Select the **commons-dbutils** package and its latest version.
4. Copy the Maven dependency record into the pom.xml file.
5. Replace version identifier with property in the same fashion the other dependencies are done.

Now we can use the `DbUtils` class for closing the resources:

```
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet set = null;

    try {

        conn = DriverManager.getConnection("some url");
        stmt = conn.prepareStatement("some query");
        set = stmt.executeQuery();

        //process results

    } catch (SQLException e) {
        //handle exception
    } finally {
        DbUtils.closeQuietly(conn, stmt, set);
    }
```

JDK 7 and Later

JDK 7 has introduced so-called try-with-resources statement.
The new language construct can be used to call the `close()` method of `Closeable` objects implicitly at the end
of the try-catch block execution.

```
    try (Connection conn = DriverManager.getConnection("some url");
        PreparedStatement stmt = conn.prepareStatement("some query");
        ResultSet set = stmt.executeQuery();) {

        //process results

    } catch (SQLException e) {
        //handle exception
        // this is run after the resources have been closed
    } finally {
        // this is run after the resources have been closed
    }
```

Working with Prepared Statements

There are several good reasons for using *PreparedStatement*.

```
    PreparedStatement stmt = conn.prepareStatement("SELECT * FROM users WHE
RE user_id = :?");
    stmt.setLong(1, 331);
```

Questions

1. How would you solve the parameter passing when using *try-with-resource*?

License

Base of the JPA setup has been created by Karel Zibar during one of the courses at the University.