Datenbanken und Web-Techniken
Exercise 2:
JDBC and JSP

JDBC
Java Database Connectivity

- Database API for Java
- Allows uniform data access to databases from different companies
- Focus on relational DBMS
- Will be presented in the lecture at a later point
  - we use it now and explore the details later
- Some links to more Information:
  - German Wikipedia: https://de.wikipedia.org/wiki/Java_Database_Connectivity
  - English Wikipedia: https://en.wikipedia.org/wiki/Java_Database_Connectivity
  - Oracle Documentation: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

Basic usage of JDBC

To access the database you need the following Java objects:

- A connection object: java.sql.Connection

```
Connection connection = DriverManager.getConnection
("jdbc:postgresql://ServerName/DatabaseName, UserName, Password");
```

- A statement object: java.sql.Statement

```
Statement statement = connection.createStatement();
statement.executeQuery("SELECT * FROM TableName");
statement.executeUpdate("UPDATE TableName SET name = 'Meier' WHERE
name = 'Schulze'");
```

- A result object: java.sql.ResultSet

```
ResultSet result = statement.executeQuery("SELECT * FROM
TableName");
```

JSP
JavaServer Pages

- Programming language for dynamic web pages in Java
- Java code and special JSP actions may be embedded in HTML and XML pages
- Requires a web server with a servlet container (like Apache Tomcat)
- Will be presented in the lecture at a later point
  - we use it now and explore the details later
- Some links to more Information:
  - German Wikipedia: https://de.wikipedia.org/wiki/JavaServer_Pages
  - English Wikipedia: https://en.wikipedia.org/wiki/JavaServer_Pages
  - Oracle Overview: https://www.oracle.com/technetwork/java/javaee/jsp/

Tasks
Hints and more description follow on the following pages

Preparation Task:

1. Import the provided pizzen.sql into your PostgreSQL database
2. Install Java Development Kit and NetBeans IDE with Java EE and Apache Tomcat Server

JDBC Task:

3. Load the provided JDBCTest-Project in your IDE
4. Modify JDBCTest.java to use your database and table
5. Compile and run the project and see the data from your table in the console log

JSP Task:

6. Load the provided PizzaJDBC-Project in your IDE
7. Compile and run the project and see a website with static pizza information
8. Modify the project to load the pizza data from your database instead of static pizza definition
9. Compile and run the project and see a website with dynamic pizza information from your database

Preparation Task 1: Import the provided pizzen.sql into your PostgreSQL database

- If you still don't have a PostgreSQL database, follow the steps of Preparation Task 1 from Exercise 1
- Follow the steps of Preparation Task 2 from Exercise 1 to get access to your database
  - please remember that you need access to the network of the TU Chemnitz, if you are using the service from URZ (use VPN from outside)
- Execute the SQL queries from the provided **`pizzen.sql`** in your database
  - this will create a new table named **`pizzen`** and adds eleven pizzas
  - you can just copy and past the lines into some SQL-console window in your used GUI
    - in phpPgAdmin this is called SQL in the upper right corner

Preparation Task 2: Install Java Development Kit and NetBeans IDE with Java EE and Apache Tomcat Server

- The provided Java projects were tested with JDK 8, NetBeans 8.2 and Tomcat 8.0
  - there are later versions, but they may lead to unexpected errors or other problems
    - latest Tomcat 9.0 works (all projects were tested last year)
    - latest NetBeans 11.0 might work (didn't test everything)
    - latest JDK 12 might work (didn't test everything)
  - you may also try to use different IDEs (Eclipse, …), but there is no support for this
    - Gradle projects are offered for these, but there is still no support
    - to distinguish them, there are the suffixes **`_NetBeans`** and **`_Gradle`**
- Java SE Development Kit 8 is still supported and the required minimum
  ➔https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - download and install before continuing (as NetBeans needs Java to run)
  - Oracle dropped support for JDK 8 during these exercises, so you might test later versions

Preparation Task 2: Install Java Development Kit and NetBeans IDE with Java EE and Apache Tomcat Server

- Old NetBeans 8.2 may still be downloaded as bundle with Java EE and Tomcat 8.0 included
  ➔https://netbeans.org/downloads/8.2/
  - choose appropriate bundle **`Java EE`** or **`All`**
  - while installing NetBeans, make sure to enable the Apache Tomcat Server (**`Customize…`**)
    - otherwise follow steps for additional download of Apache Tomcat
  - afterwards start NetBeans and install or activate some plugins, which are not installed by default
    - **`Java EE Base`** and **`EJB and EAR`** (both of category **`Java Web and EE`**)
    - in case of errors (especially for later version of NetBeans), just activate all **`Java Web and EE`** plugins
    - otherwise you can't open JSP projects
- Additional download of Apache Tomcat (if not downloaded and activated as bundle with old NetBeans)
  ➔https://tomcat.apache.org/download-90.cgi
  - just download and extract to some location (no installer is required)
  - in NetBeans open **`Tools → Servers → Add Server … → Apache Tomcat or TomEE`**
    - provide server location (for example **`apache-tomcat-9.0.17`** directory)
    - select a username and password for your server (you have to provide this, whenever you run a JSP project)

JDBC Task 3: Load the provided JDBCTest-Project in your IDE

- Download and extract the provided **`JDBCTest.zip`**
- Open your NetBeans IDE
- Select **`File → Open Project…`** and choose the **`JDBCTest`** folder
  - there should be a different folder icon (brown coffee pot icon or NetBeans project icon)

JDBC Task 4: Modify JDBCTest.java to use your database and table

- Select the file **`JDBCTest.java`** in the directory tree at the top left side
  - **`JDBCTest → Souce Packages → <default package> → JDBCTest.java`**
- Double click or press Enter to open in editor window at the top right side
- There are several variables / constants defined at the beginning (lines 23 to 28)
  - change them accordingly (and replace **`your_…`** with the correct values)
  - you may use the same as in exercise 1 or the new **`pizzen`** table, you just created

JDBC Task 5: Compile and run the project and see the data from your table in the console log

- Select **`Run → Run Project`** (this will save and build your project automatically)
- Console log will be shown at the bottom right side
  - in case of error messages, you have to solve them

JSP Task 6: Load the provided PizzaJDBC-Project in your IDE

- Download and extract the provided **`PizzaJDBC.zip`**
  - contains a project which is taken from a JSP tutorial from the IX magazine 7/2000 p. 152
    - http://www.heise.de/artikel-archiv/ix/2000/07/152
    - unfortunately, this is now only available with costs
- Open your NetBeans IDE
- Select **`File → Open Project…`** and choose the **`PizzaJDBC`** folder
  - there should be a different folder icon (blue globe icon or NetBeans project icon)
  - if the project is not recognized, go back to Task 2 and activate Java EE in NetBeans

JSP Task 7: Compile and run the project and see a website with static pizza information

- Select **`Run → Run Project`**
- You will get prompted for the username and password of the Tomcat server, if you didn't use the bundle
- Your firewall might also ask for permissions
- If your browser doesn't open, you have to select one
  - Select **`Tools → Options → General → Web Browser`** and choose your favourite
- In case of problems, take a look at the console log
  - there will be multiple logs, as Tomcat has its own tab
- On website press **`Bestellen`** and hopefully see some static dummy pizza data

JSP Task 8: Modify the project to load the pizza data from your database instead of static pizza definition

- Open the file **`PizzaList.java`** in the directory tree at the top left side
  - **`PizzaWeb → Souce Packages → de.ix.jspTutorial.model → PizzaList.java`**
- Go to method **`readListFromDB()`**
- Read the comments and try to understand, how the **`pizzas`** object gets created
  - especially the **`Pizza`** class is important
- Add some code, to load the pizza data from your database
  - Hint: see JDBC task for some helpful code
- Fill the **`pizzas`** object with the information from database
  - use the aid of the IDE (autocomplete, suggestions, possible methods, …)
  - it might be a good point to learn Java, as we are going to use it for the rest of the exercises

JSP Task 9: Compile and run the project and see a website with dynamic pizza information from your database

- You should know, how to run your program by now :)
- In case of problems, take a look at the console log
- On website press **`Bestellen`** and hopefully see the pizza data from your database