МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра «стратегічне управління»

ЗВІТ

з лабораторної роботи №4

з дисципліни

«ПРОГРАМУВАННЯ ВЕБ-ЗАСТОСУВАНЬ java ee»

ВИКОНАВ

Студент групи КН-317А

ПЕРЕВІРИла

ас. каф. СУ

Луценко С. Ю.

Харків 2020

ДОСЛІДЖЕННЯ ПРИНЦИПІВ СТВОРЕННЯ ВЕБ-ЗАСТОСУВАНЬ  
ДЛЯ РОБОТИ ІЗ БАЗАМИ ДАНИХ

Мета

Основна мета – навчитись створювати консольні та веб-застосування, що відображають на веб-сторінках інформацію, яка міститься у базі даних.

1. Набуття навиків підключення БД до Java-проекту та реалізації CRUD на основи технології JDBC.
2. Вдосконалення навиків створення веб-сторінок на основні технології JSP.
3. Вдосконалення навиків застосування JSTL при розробці JSP.
4. Вдосконалення навиків розробки userfrendlyapplications.
5. Вдосконалення навиків створення сервлетів, враховуючи особливості використання даних.
6. Набуття навиків реалізації контроля коректності даних, що вводяться до бази даних.

Завдання роботи

**Завдання 1.** Дослідження готового консольного застосування, що працює із БД.

**Завдання 2.** Розробити консольне застосування, яке виконує підключення та обробку даних, що містяться в реляційній базі даних, аналогічно додатку, розглянутому в завданні 1, але для власної предметної області. Рекомендуємий формат бази данихMySQL.

Для виконання цього завдання рекомендується застосовувати шаблон, представлений у вигляді проекту TestLR3\_Pattern.

**Завдання 3.** Розробити веб-застосування, яке працює із данимиз однією таблицею реляційної базы даних, яке дозволяє:

- відобразитивміст таблиці на веб-сторинці;

- додавати, редагувати та вилучати інформацію з таблиці БД;

- здійснювати пошук даних за семантичними полями.

Проект розроблюється відповідно до обраної предметної області. При його створенні пропонується використовувати проект, отриманий у результаті виконання ЛР №3.

Хід виконання

**Завдання 1.** Дослідження готового консольного застосування, що працює із БД.

Консольний за стосунок було вдало запущено за допомогою стандартної консолі Windows 7 – рисунок 1-2.

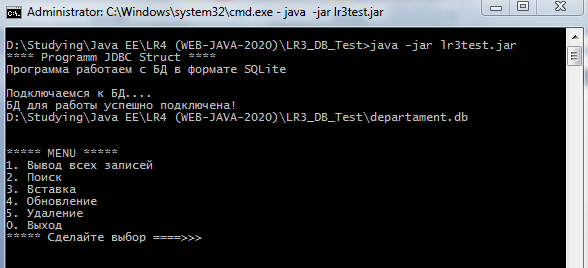


Рисунок 1 – API консольного застосунку

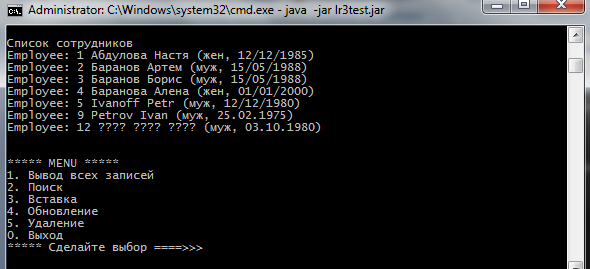


Рисунок 2 – Використання консольного застосунку

**Завдання 2.** Розробити консольне застосування, яке виконує підключення та обробку даних, що містяться в реляційній базі даних.

Для виконання цього завдання був використаний шаблон TestLR3\_Pattern. До BuildPath був доданий драйвер JDBC для з’єднанням з базою даних MySql. Структуру проекту можна побачити на рисунку 3.

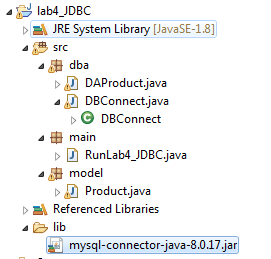


Рисунок 3 – Структура проету lab4\_JDBC

Лістинг видозміненого коду наведено у лістингу 1.

Лістинг 1 – файл RunLab4\_JDBC.java

|  |
| --- |
| package main;  import dba.DAProduct;  import dba.DBConnect;  import model.Product;  import java.util.Scanner;  public class RunLab4\_JDBC {  public static void main(String[] args) {  System.out.println("\*\*\*\* Programm JDBC Struct \*\*\*\*");  Scanner scan = new Scanner(System.in);  DBConnect.DBConection("jdbc:mysql://localhost:3306/java?useUnicode=true&characterEncoding=UTF8", "root", "\*\*\*\*\*\*");  Product product;    //Действия с БД  //Меню!!!  int choice = 0;  do {  choice = menu(scan);  switch (choice) {  case 1:  //SELECT  DAProduct.showAll(DBConnect.getConnection(), "Список продуктов");  break;  case 2:  System.out.println("Поиск продукта");  DAProduct.seek(DBConnect.getConnection(), Product.createProductForSearching(scan));  break;  case 3:  //INSERT  …  case 4:  //UPDATE  …  case 5:  //DELETE  …  case 0:  System.out.println("\n");  break;  default:  System.out.println("Неправильный выбор");  }  } while (choice != 0);  DBConnect.DBClose();  scan.close();  System.out.println("\*\*\*\* THE END \*\*\*\*");  }  public static int menu(Scanner scan) {  …  }  } |

**Завдання 3.** Розробити веб-застосування, яке працює із даними з однією таблицею реляційної базы даних.

Структуру проекту наведено на рисунку 4.

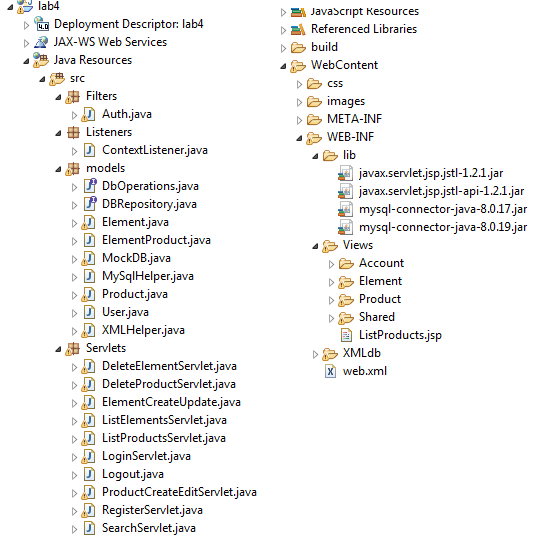


Рисунок 4 – Структура проекту lab4

Був створений новий клас для роботи з базою даних MySqlHelper.java замість XMLHelper.java. Він реалізує той самий інтерфейс DbOperations.java, що й XMLHelper, у якому визначені методи для роботи з бд - стандартні операції CRUD. Реалізація цього інтерфейсу дозволяє не змінювати код у сервлетах, а тільки наділити їх певним об’эктом інтерфейсу DbOperations.java. Цей механізм (Dependency Injection) дозволить використовувати будь-яку базу даних на вибір. При цьому кожен сервлет може використовувати власну бд. Тип бд для сервлетів визначається тільки в одному файлі – ContextListener.java, метод contextInitialized() якого викликається один раз на старті серверу. Код цього класу наведено у лістингу 2.

Лістинг 2 - ContextListener.java

|  |
| --- |
| package Listeners;  import javax.servlet.ServletContextEvent;  import javax.servlet.ServletContextListener;  import javax.servlet.annotation.WebListener;  import Servlets.DeleteElementServlet;  import Servlets.DeleteProductServlet;  import Servlets.ElementCreateUpdate;  import Servlets.ListElementsServlet;  import Servlets.ListProductsServlet;  import Servlets.LoginServlet;  import Servlets.ProductCreateEditServlet;  import Servlets.RegisterServlet;  import Servlets.SearchServlet;  import models.DbOperations;  import models.MySqlHelper;  import models.XMLHelper;  @WebListener  public class ContextListener implements ServletContextListener{    @Override  public void contextInitialized(ServletContextEvent event) {  DbOperations xmlHelper = new XMLHelper();  DbOperations mySqlHelper = new MySqlHelper();    System.out.println("Adding xmlHelper to servlets managing everything except products");  DeleteElementServlet.setDbHelper(xmlHelper);  ElementCreateUpdate.setDbHelper(xmlHelper);  ListElementsServlet.setDbHelper(xmlHelper);  LoginServlet.setDbHelper(xmlHelper);  RegisterServlet.setDbHelper(xmlHelper);  System.out.println("Adding mySqlHelper to servlets managing products");  SearchServlet.setDbHelper(mySqlHelper);  DeleteProductServlet.setDbHelper(mySqlHelper);  ListProductsServlet.setDbHelper(mySqlHelper);  ProductCreateEditServlet.setDbHelper(mySqlHelper);  }  } |

Об’єкт класу MySqlHelper.java створюється один раз і при його створенні здійснюється з’єднання з базою даних. Для виконання завдання методи CRUD були створені для моделі Product. Код MySqlHelper.java наведено у лістингу 3.

Лістинг 3 - MySqlHelper.java

|  |
| --- |
| package models;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.List;  import models.Product;  public class MySqlHelper implements DbOperations{  private static Connection con;    public static Connection getConnection() {  return con;  }  public MySqlHelper() {  try {  System.out.println("Connecting to DB");  Class.forName("com.mysql.jdbc.Driver");  con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java", "root", "\*\*\*\*\*\*");  System.out.println("DB connected");    } catch (SQLException | ClassNotFoundException ex) {  ex.printStackTrace();  System.out.println("Error connecting to DB");  System.exit(222);  }  }  @Override  // CRUD methods…  } |

Для пошуку продуктів був створений новий сервлет SearchServlet.java. Його код наведено у лістингу 4.

Лістинг 4 - SearchServlet.java

|  |
| --- |
| package Servlets;  import java.io.IOException;  import java.util.stream.Collectors;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import models.DbOperations;  @WebServlet("/Search")  public class SearchServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private static DbOperations dbHelper = null;  public static void setDbHelper(DbOperations DbHelper) {  dbHelper = DbHelper;  }  public SearchServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.getWriter().append("Served at: ").append(request.getContextPath());  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  String text\_to\_find = request.getParameter("text\_to\_find");    request.setAttribute("products", dbHelper.getProducts().stream().filter(el -> el.getName().contains(text\_to\_find)).collect(Collectors.toList()));  request.getRequestDispatcher("WEB-INF/Views/Product/ListProducts.jsp").forward(request, response);  doGet(request, response);  }  } |

Бібліотека JSTL використовувалася вже раніше, але задля виконання умов завдання нижче наведений приклад її використання – лістинг 5.

Лістинг 5 – Приклад розмітки з файлу ListProducts.jsp

|  |
| --- |
| <div class=*"container"*>  <div class=*"row"*>  <c:forEach items = *"*${ products }*"* var = *"product"*>  <div class=*"col-xs-12 col-sm-6 col-md-4"*>  <div class=*"card h-100"*>  <div class=*"card-body bg-light text-center"*>  <p><a href = *"*${pageContext.request.contextPath}*/Products?id=*${ product.getId() }*"*><img class=*"img-product"* src=*"*${pageContext.request.contextPath}*/images/*${ product.getEng\_name() }*"*  height = *"120"*></a></p>  <h4 class=*"card-title"*>${product.getName()}</h4>  <p class=*"card-text"*>  <%-- <c:forEach items = "${ product.getElementProducts() }" var = "elementProduct"> --%>  Белки : ${ product.getProteins() }  <br>  Жиры : ${ product.getFats() }  <br>  Углеводы : ${ product.getCarbohydrates() }  <!-- <br> -->  <%-- </c:forEach> --%>  </p>  <p class=*"card-footer bg-light"*>  <a class=*"btn btn-info"* href = *"*${pageContext.request.contextPath}*/CreateUpdateProduct?id=*${ product.getId() }*"*>Править</a>  <a onclick = "deleteProduct(this)" id = *"*${ product.getId() }*"* class=*"btn btn-danger"* href = *"#"*>Удалить</a>  </p>  </div>  </div>  </div>  </c:forEach>  </div>  </div> |

Приклад перевірки полів на стороні клієнта зображено на рисунку 5.

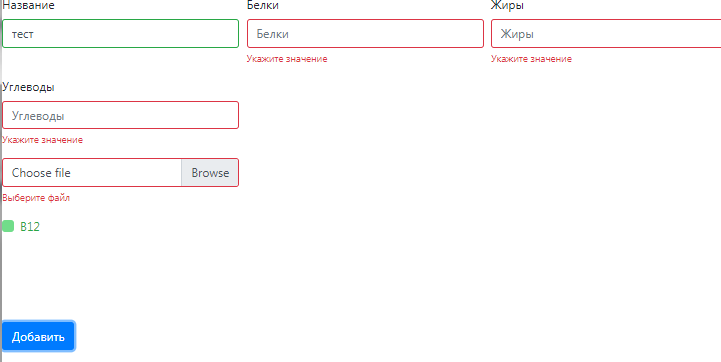


Рисунок 5 – Перевірка полів

Приклад роботи форми пошуку зображено на рисунку 6.

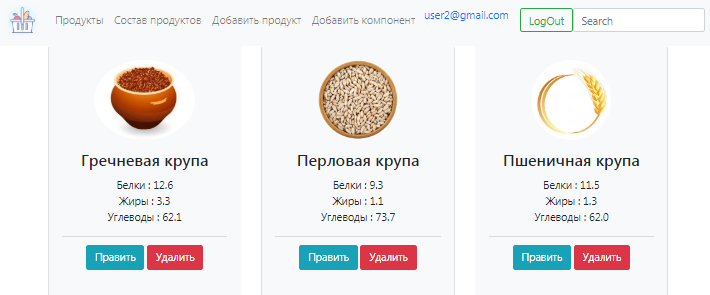
 

Рисунок 6 – Пошук продуктів, що мають у собі слово «крупа». Адреса змінюється на <http://localhost:8080/lab3/Search>

Первинні дані до бази додавалися за допомогою функцій mysql.exe у консолі. Приклад sql запитів наведено у лістингу 6.

Лістинг 6 – Приклад sql запитів

|  |
| --- |
| create table products (id int auto\_increment, name varchar(50), eng\_name varchar(50), carbohydrates double, proteins double, fats double, primary key(id));  create table elements (id int auto\_increment, name varchar(50), primary key(id));  create table elementproducts (product\_id int, element\_id int, primary key(product\_id, element\_id));  insert into products values(null, 'Хлеб пшеничный', 'bread.png', 48.8, 8.1, 1.0); |

Висновки

При виконанні цієї лабораторної роботи був здійснений важливий крок в розробці веб-застосування, а саме з’єднання з базою даних. У ролі бази даних була обрана MySql, бо вона має багато можливостей, що потенційно можуть знадобитися в майбутній розробці. Були додані первинні дані до бази за допомогою консольного api. Були додані методи для роботи з базою у додатку.