ЛАБОРАТОРНА РОБОТА 5

Тема: Реалізація баз даних у корпоративних додатках Java
Мета: Створення web-додатку, що використовує базу даних

Лабораторна робота передбачає використання IDE NetBeans з активованою підтримкою Jakarta EE/JavaEE та сервером додатків GlassFish. Для її виконання необхідно додатково встановити IDE NetBeans (https://netbeans.apache.org/).

Практична частина

Постановка задачі

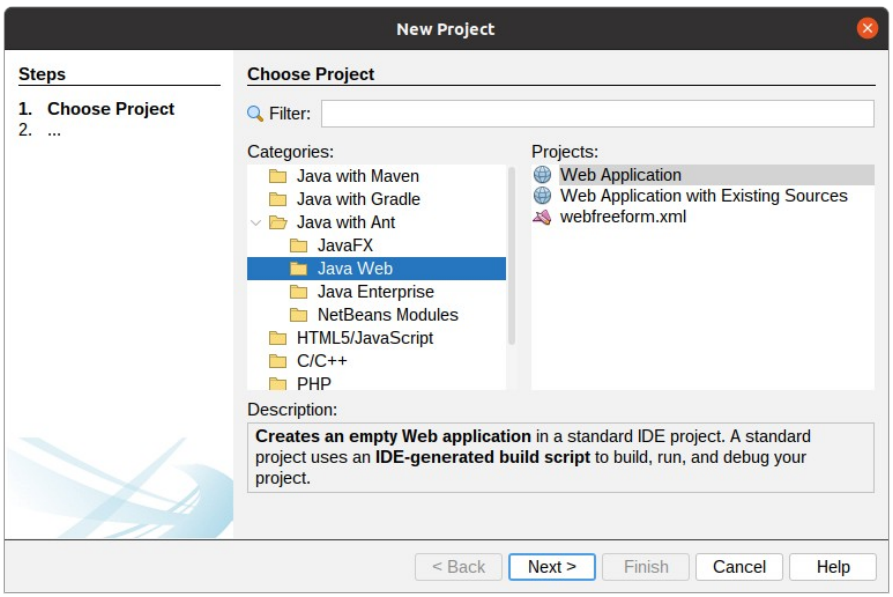
Необхідно розробити web-додаток, який використовує сервлет для пошуку інформації про співробітників організації. Дані про співробітників зберігаються у таблиці Employee. Для реалізації пошуку користувач вказує прізвище співробітника. Результатом пошуку є інформація про співробітників, записи про яких має база даних (можливо існування декількох співробітників з однаковими прізвищами).

Для рішення наведеної задачі необхідно виконати наступні кроки:

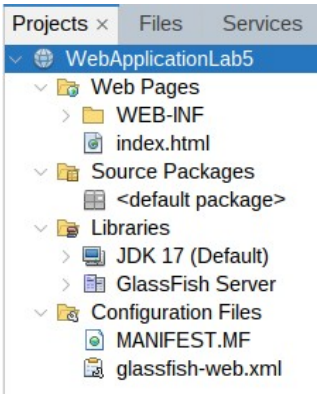
1. Створити новий проєкт.
2. Створити таблицю `employee` та заповнити її даними.
3. Розробити сервлет, який вибирає з БД записи, які відповідають запиту користувача та показує результат.
4. Запакувати додаток у war-файл та розгорнути його на сервері.
5. Виконати тестування роботи додатку з використанням браузеру.

Створення нового проєкту

1) Оберіть пункт меню File/New project, у вікні, що з’явиться, слід вказати тип проєкту Java Web/Web Application та натиснути Next.

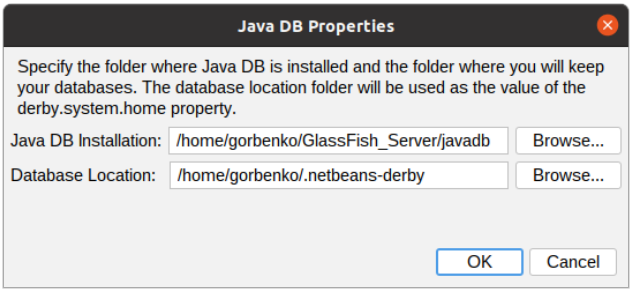


2) При необхідності можна змінити ім’я проєкту (WebApplicationLab5) та натиснути Next. Наступним можна уточнити налаштування сервера, на якому буде розгортатись web-додаток та платформу Java, натиснути Finish.
3) У результаті буде створено проєкт із наступною структурою:

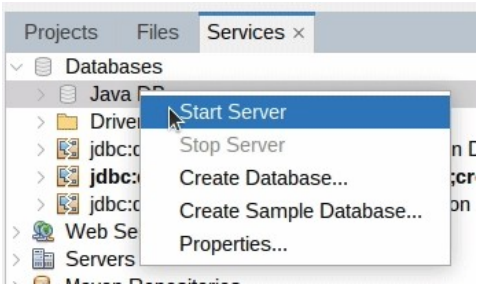


Створення таблиці `employee` і додавання тестових даних

1) Перевіряємо і за необхідністю змінюємо налаштування JavaDB



2) Запускаємо сервер бази даних



Якщо запуск був вдалим, то в консолі повідомлень з’явиться

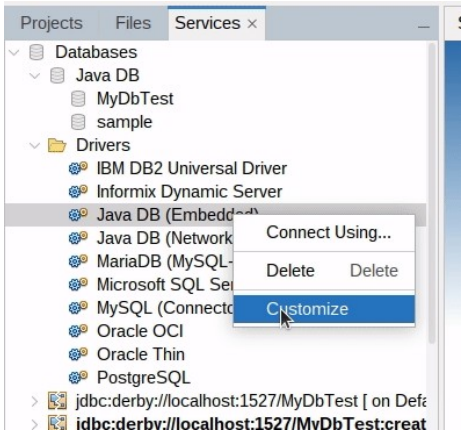
```
Сетевой сервер Apache Derby Network Server - 10.16.1.1 - (1901046) запущен и готов принимать соединения на порту 1527
```

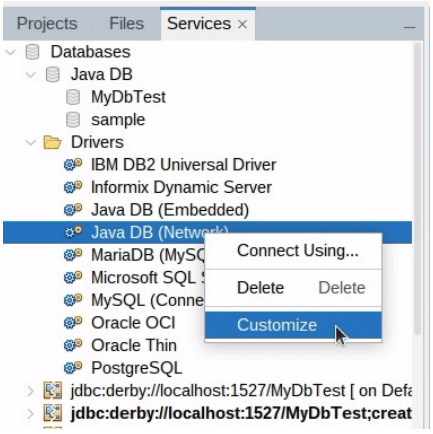
Якщо запуск серверу баз даних (Derby DB) не відбувся, то слід перевірити наступні налаштування:

- повинно бути встановлено значення змінної оточення JAVA_HOME (наприклад, JAVA_HOME=/opt/jdk17);
- повинно бути додано до значення змінної оточення PATH розташування JDK bin (наприклад, PATH=$JAVA_HOME/bin:$PATH)
- повинно бути встановлено значення змінної оточення DERBY_INSTALL (наприклад, DERBY_INSTALL=/opt/Apache/db-derby-10.16.1.1-bin)
- повинно бути встановлено значення змінної оточення CLASSPATH (наприклад, CLASSPATH=$DERBY_INSTALL/lib/derby.jar:$DERBY_INSTALL/lib/derbytools.jar: $DERBY_INSTALL/lib/derbyoptionaltools.jar:$DERBY_INSTALL/lib/derbyshared.jar:).

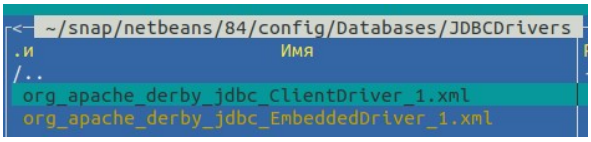
Також необхідно перевірити налаштування в IDE NetBeans. Для DerbyDB версії 10.16.1.1 змінено розташування класів JDBC — тепер вони упаковані у файл derbytools.jar. Тому необхідно внести

зміни в налаштування драйверів за допомогою діалогових вікон Java DB (Embedded) та Java DB (Network)

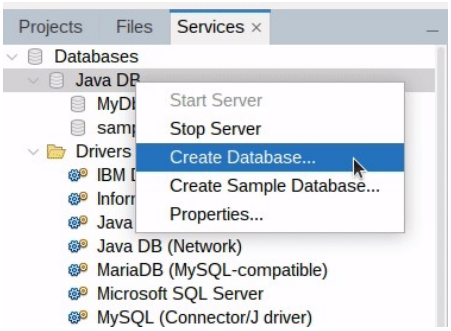


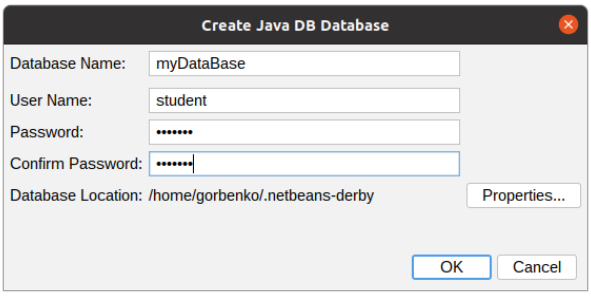


або через безпосереднє редагування файлів

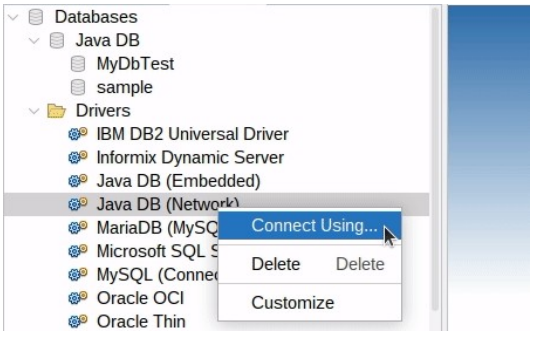


3) Створюємо нову базу даних

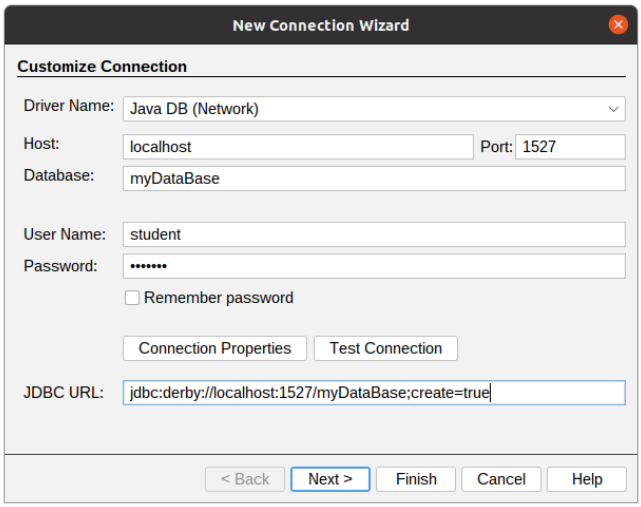




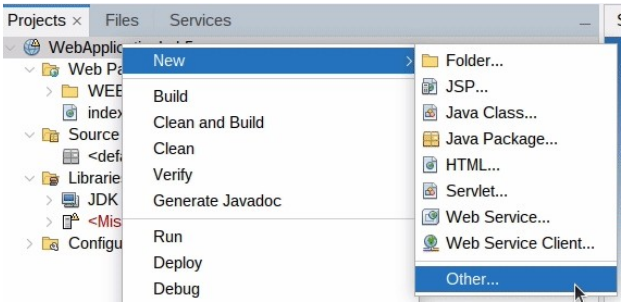
Альтернативно базу даних можна створити через команду до серверу Derby DB. Наприклад, в IDE NetBeans через відповідний драйвер робимо виклик

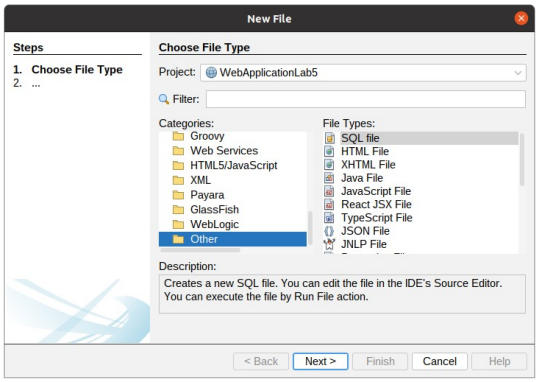


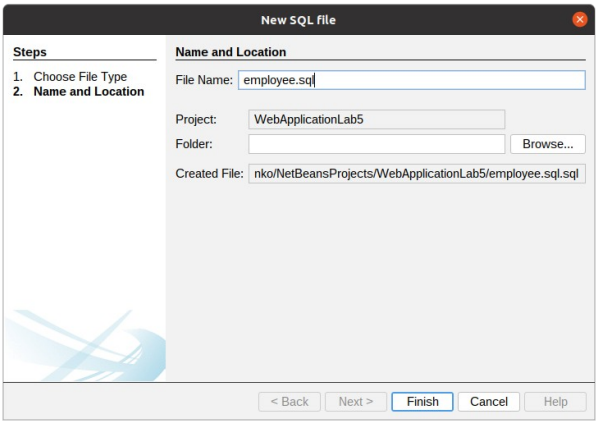
і у створеному діалоговому вікні задаємо параметри серверу Derby DB (його адресу та порт), а також ім’я бази даних та логіну доступу до неї. Для створення бази даних додаємо команду create=true.



4) Для зберігання SQL-скриптів додамо новий файл employee.sql.





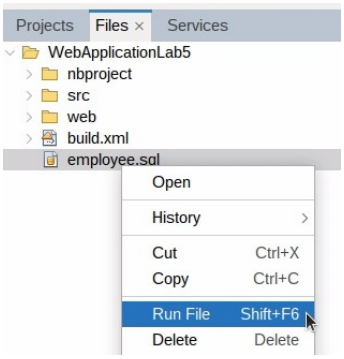


5) Створений файл автоматично відкривається для редагування. Скопіюйте у файл наступні команди:

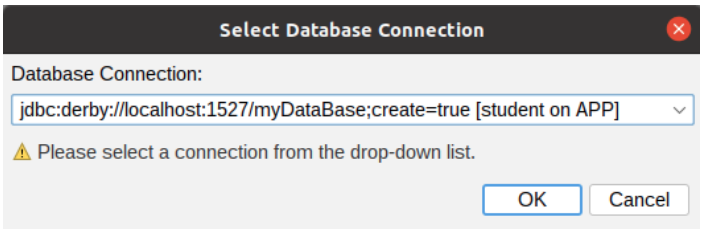
```
-- створення таблиці
create table employee(id integer, first_name varchar(20), last_name varchar(20),
designation varchar(20), phone varchar(20));

--додавання тестових даних
insert into employee values (1, 'Ivan', 'Petrenko', 'Manager', '11-22-33');
insert into employee values (2, 'Mykola', 'Kozhedub', 'Programmer', '33-44-55');
insert into employee values (3, 'Sergii', 'Zinchenko', 'System administrator', '12-
34-56');
```

```
insert into employee values (4, 'Oksana', 'Krutova', 'Manager', '56-78-90');
insert into employee values (5, 'Oleksii', 'Kuznetsov', 'Technician', '55-66-77');

-- обрати все з таблиці для перевірки
select * from employee;
```

6) Збережіть файл.
7) Клацніть правою кнопкою миші на файл employee.sql та оберіть Run file (виконати)



8) Оберіть базу даних, до якої необхідно під’єднатись при виконані SQL команд

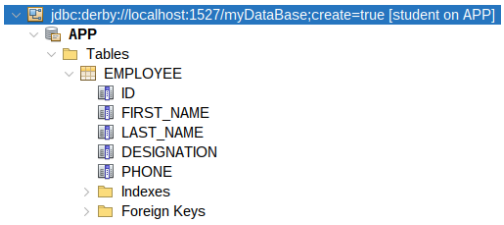


9) При вдалому виконані команд скрипту будуть виводитись відповідні повідомлення та створяться таблиці бази даних, до яких заносяться дані

select * from employee ×

Max. rows: 100 Fetched Rows: 5

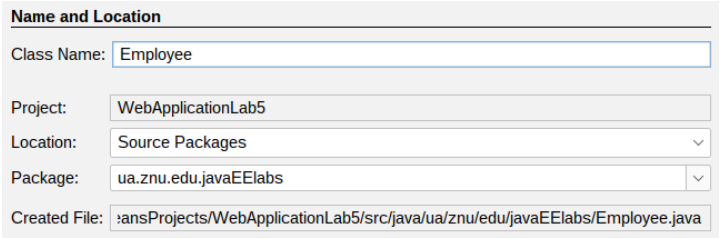
#	ID	FIRST_NAME	LAST_NAME	DESIGNATION	PHONE
1	1	Ivan	Petrenko	Manager	11-22-33
2	2	Mykola	Kozhedub	Programmer	33-44-55
3	3	Sergii	Zinchenko	System administrator	12-34-56
4	4	Oksana	Krutova	Manager	56-78-90
5	5	Oleksii	Kuznetsov	Technician	55-66-77



Реалізація сервлету

Перед розробкою сервлету, створимо звичайний Java клас Employee, який буде використовуватись для представлення та передавання даних про співробітників.

1) Клацаємо правою кнопкою миші на проєкт WebApplicationLab5 і обираємо пункт меню New -> Java class

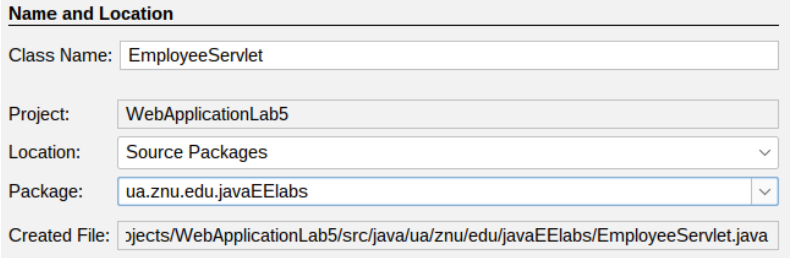


2) У класі `Employee` створюємо п’ять полів, які відповідають стовпчикам таблиці `employee`, додаємо конструктори і набір get/set методів. Повний код класу `Employee` наведено нижче:

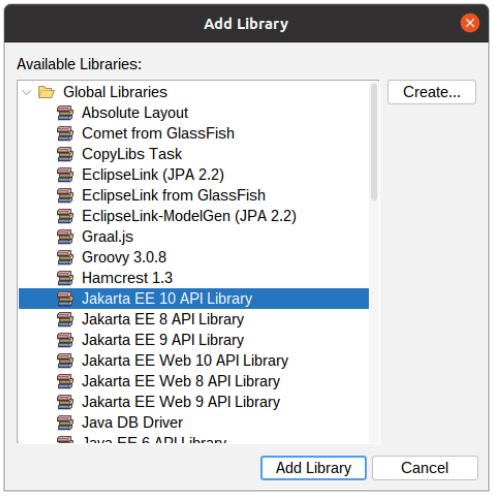
```
package ua.znu.edu.javaEElabs;

import java.io.Serializable;

public class Employee implements Serializable {

	private Long id;
	private String firstName;
	private String lastName;
	private String designation;
	private String phone;

	public Employee() {}

	public Employee(Long id, String firstName, String lastName,
			String designation, String phone) {
		super();
		this.id = id;
		this.firstName = firstName;
		this.lastName = lastName;
		this.designation = designation;
		this.phone = phone;
	}

	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getDesignation() {
		return designation;
	}
	public void setDesignation(String designation) {
		this.designation = designation;
	}
```

```
	public String getPhone() {
		return phone;
	}
	public void setPhone(String phone) {
		this.phone = phone;
	}
}
```

3) Для створення нового сервлету клацаємо правою кнопкою миші на проєкт та обираємо пункт меню New -> Servlet. Задаємо ім'я класу EmployeeServlet та обираємо пакет, в якому буде розміщено сервлет.



4) У створеному шаблоні сервлету використовується імпорт класів із пакету javax.servlet. Якщо створюється додаток на основі технологій JakartaEE, то необхідно змінити ці посилання на jakarta.servlet. Крім того необхідно додати до проєкту відповідну бібліотеку, для чого клацаємо правою кнопкою миші на Libraries і обираємо Add Library. Серед запропонованих варіантів обираємо потрібний, наприклад, JakartaEE 10 API Library:



5) Необхідний код сервлету представлено нижче. Основну бізнес логіку реалізовано в методі `doGet()`. Зробіть копію коду і збережіть сервлет.

```
package ua.edu.znu.javaEElabs.WebApplicationLab5;

import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.ArrayList;
```

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;

@WebServlet(name = "EmployeeServlet", urlPatterns = {"/EmployeeServlet"})
public class EmployeeServlet extends HttpServlet {

    public EmployeeServlet() {
       super();
         }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
   try {
       response.setContentType("text/html;charset=UTF-8");
       // Отримання з http-запиту значення параметру lasname
       String lastname = request.getParameter("lastname");

       // Колекція для зберігання знайдених співробітників
       ArrayList<Employee> employees = new ArrayList<Employee>();

       // Завантаження драйверу Derby DB
        Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

       // Отримання з'єднання з БД
        Connection con = DriverManager.getConnection(
                "jdbc:derby://localhost:1527/myDB;create=true;user=student;password=student");

       // Виконання SQL-запиту
       ResultSet rs = con.createStatement().executeQuery(
                "Select id, first_name, last_name, designation, phone "
                + "From employee " + "Where last_name like '"
              + lastname + "'");
       // Пересилання результатів запиту
       while (rs.next()) {
              // По кожному запису вибірки формується
              // об'єкт класу Employee.
              // Значення властивостей заповнюються із полів запису
              Employee emp = new Employee(
                         rs.getLong(1),
                         rs.getString(2),
                         rs.getString(3),
                         rs.getString(4),
                         rs.getString(5));
              // Додавання створеного об'єкту до колекції
              employees.add(emp);
                 }
       // Закриваємо вибірку та з'єднання з БД
       rs.close();
       con.close();

       // Виводимо інформацію про знайдених співробітників
       PrintWriter out = response.getWriter();
       out.println("Знайдені співробітники<br>");
       for (Employee emp: employees) {
              out.print(emp.getFirstName() + " " +
                           emp.getLastName() + " " +
                           emp.getDesignation() + " " +
                           emp.getPhone() + "<br>");
       }
    } catch (Exception ex) {
       ex.printStackTrace();
       throw new ServletException(ex);
    }
}

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

```
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Тестування додатку

Запустіть браузер і перейдіть за посиланням

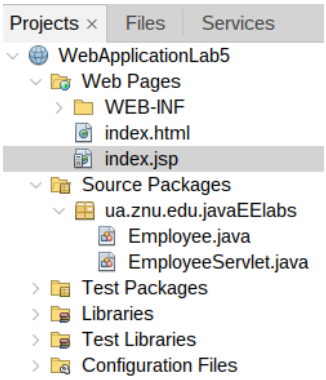
http://localhost:8080/WebApplicationLab5/EmployeeServlet?lastname=Petrenko

Зверніть увагу, що у запиті виконується звернення до сервлету і передається параметр lastname із значенням Petrenko. Зробіть скриншот отриманим результатам запиту.

Розробка JSP-сторінки

Для покращення функціональності додатку використовуються JSP-сторінки. Додамо до проєкту одну сторінку index.jsp, яка буде використовуватись для введення прізвища і відображення результатів пошуку.

1) Для створення нової JSP сторінки, натисніть правою кнопкою миші на проєкт, далі — New/JSP. У відкритому діалоговому вікні вкажіть ім’я файлу index та натисніть Готово.



До проєкту було додано файл index.jsp. Для того, щоб цей файл використовувався автоматично необхідно із проєкту видалити файл index.html.

2) Створений JSP файл вже вміщує базовий набір тегів заголовку та тіло веб-сторінки. Для рішення поставленої задачі необхідно внести ряд змін:
   - змінити заголовок сторінки на «Пошук співробітників»
   - створити форму, яка має текстове поле lastname для введення прізвища та кнопку підтвердження (Submit). При виконані підтвердження форма передає значення параметру `lastname` сервлету `EmployeeServlet`.
   - сигналом того, що пошук був виконаний та є результати для відображення, буде слугувати наявність параметру http-запиту `employeesFound`, який буде передаватись сервлетом до сторінки index.jsp у випадку успішного виконання пошуку. Перевірка розміру колекції знайдених співробітників дозволяє визначити чи було, принаймні, одного співробітника додано. У випадку позитивної відповіді оброблюємо колекцію і відображаємо усі властивості кожного знайденого співробітника у вигляді таблиці.

Код JSP-сторінки:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@page import="java.util.ArrayList"%>
<%@page import="ua.edu.znu.javaEElabs.WebApplicationLab5.Employee"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Пошук співробітників</title>
</head>
<body>
<form action="EmployeeServlet">
      Прізвище співробітника
      <input type="text" name="lastname">
      <input type=submit value="пошук">
</form>
<%
// отримання значення параметру employeesFound
ArrayList employees = (ArrayList)
request.getAttribute("employeesFound");
// Якщо параметр задано, починаємо обробку
if (employees != null) {
      // Якщо не знайдено ні одного співробітника - вивід повідомлення
      if (employees.size()==0)
            out.print("Співробітники не знайдено");
      else {
      out.print("<TABLE border=\"1\">");
      // Заголовок таблиці
      out.print("<TR><TD>Id</TD><TD>Ім’я</TD><TD>Прізвище</TD>" +
                  "<TD>Посада</TD><TD>Телефон</TD></TR>");
      for (int i = 0; i < employees.size(); i++) {
            // По кожному знайденому співробітнику
            // формується рядок таблиці
            out.print("<TR>");
            // Отримання чергового співробітника з колекції
            Employee emp = (Employee) employees.get(i);
            // Заповнення рядку таблиці даними співробітника
            out.print("<TD>" + emp.getId() + "</TD>");
            out.print("<TD>" + emp.getFirstName() + "</TD>");
            out.print("<TD>" + emp.getLastName() + "</TD>");
            out.print("<TD>" + emp.getDesignation() + "</TD>");
            out.print("<TD>" + emp.getPhone() + "</TD>");
            out.print("</TR>");
      }
      out.print("</TABLE>");
      }
}
%>
</body>
</html>
```

Доопрацювання сервлету

Метод сервлету `doGet()` виконується після того, як користувач виконав підтвердження (Submit) форми. Необхідно змінити код сервлету таким чином, щоб результати пошуку відправлялись jsp-сторінці у вигляді параметру `employeesFound`. Для цього закоментуємо частину коду, яку пов’язано з виведенням колекції співробітників у потік виводу (`response.getWriter()`), і додамо код розміщення цієї колекції до параметру запиту і перенаправлення запиту до сторінки index.jsp:

```
/*
// Виводимо інформацію про знайдених співробітників
PrintWriter out = response.getWriter();
out.println("Знайдено співробітників<br>");
```

```
for (Employee emp: employees) {
      out.print(emp.getFirstName() + " " +
                  emp.getLastName() + " " +
                  emp.getDesignation() + " " +
                  emp.getPhone() + "<br>");
}
*/

// Розміщення результатів до параметру запиту employeesFound
request.setAttribute("employeesFound", employees);
// Перенаправлення http-запиту на сторінку index.jsp
RequestDispatcher dispatcher = getServletContext()
            .getRequestDispatcher("/index.jsp");
dispatcher.forward(request, response);
```

Тестування додатку

Запустіть браузер і перейдіть за адресою http://localhost:8080/WebApplicationLab5/.

Уведіть прізвище співробітника і натисніть «Пошук». Зробіть скриншот отриманого результату. Яким буде результат пошуку, якщо залишити поле «Прізвище співробітника» незаповненим.

Завдання

1. Використовуючи IDE NetBeans виконайте надану практичну частину і впевніться у її роботі. Результат надайте у звіті.
2. Використовуючи матеріал попередніх лабораторних робіт виконайте реалізацію практичної частини в IDE Eclipse для сервера додатків WildFly та СУБД Derby DB. Реалізацію, з відповідними скриншотами, надайте до звіту.
3. Розширте функціональні можливості функцією додавання нових співробітників. Створять нову сторінку, яка має форму для введення інформації про нових співробітників та додайте до неї посилання на головну сторінку. Необхідно розробити сервлет для обробки параметрів нового співробітника та створення запису у БД. Після створення запису нового співробітника головна сторінка повинна автоматично оновлюватись.
4. Додайте функціональні можливості додатку - функцію редагування параметрів співробітника. Необхідно створити нову сторінку, яка вміщує форму для зміни інформації про співробітника. Сторінка повинна відкриватись при клацанні мишкою по запису про співробітника у таблиці. Розробити сервлет для обробки параметрів співробітників та оновлення запису у БД. Після зміни атрибутів співробітника головна сторінка повинна автоматично оновлюватись.
5. Підготуйте та надайте звіт виконання лабораторної роботи.

Контрольні запитання

1. Де в ієрархії класів знаходиться клас HttpServlet ?
2. Чому для наведеного коду сервлету не можна використовувати версію JDK 8 ? Що слід змінити в його коді, якщо знадобиться реалізація саме для цієї версії JDK ?
3. При використанні jsp яку частину його коду буде отримувати браузер на боці клієнта? Чим це регулюється?