МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ульяновский государственный технический университет»

Ю.А. Лапшов

«Публикация баз данных в Интернете»

Методические указания по выполнению лабораторных работ

Ульяновск
УлГТУ
2016

Оглавление

Лабораторная работа №1. Установка и тестирование компонентов программного обеспечения

Цель работы: В данной лабораторной работе требуется осуществить выполнение установки и тестирование следующих программных компонентов, которые будут использованы при выполнении последующих лабораторных работ:

- Java development kit
- Среда разработки Eclipse
- Сервер приложений WildFly (в более ранних версиях - JBoss)
- JBoss Tools
- СУБД PostgreSQL

Выполнение работы:

Установка JDK.

Для установки JDK требуется скачать его дистрибутив с официального сайта Oracle http://www.oracle.com и установить с настройками по умолчанию.

Установка Eclipse

Для установки Eclipse удобнее всего воспользоваться инструментом Eclipse Installer, доступным по ссылке http://www.eclipse.org/downloads

После запуска Eclipse Installer в мастере установки следует выбрать версию Eclipse IDE for Java EE Developers:

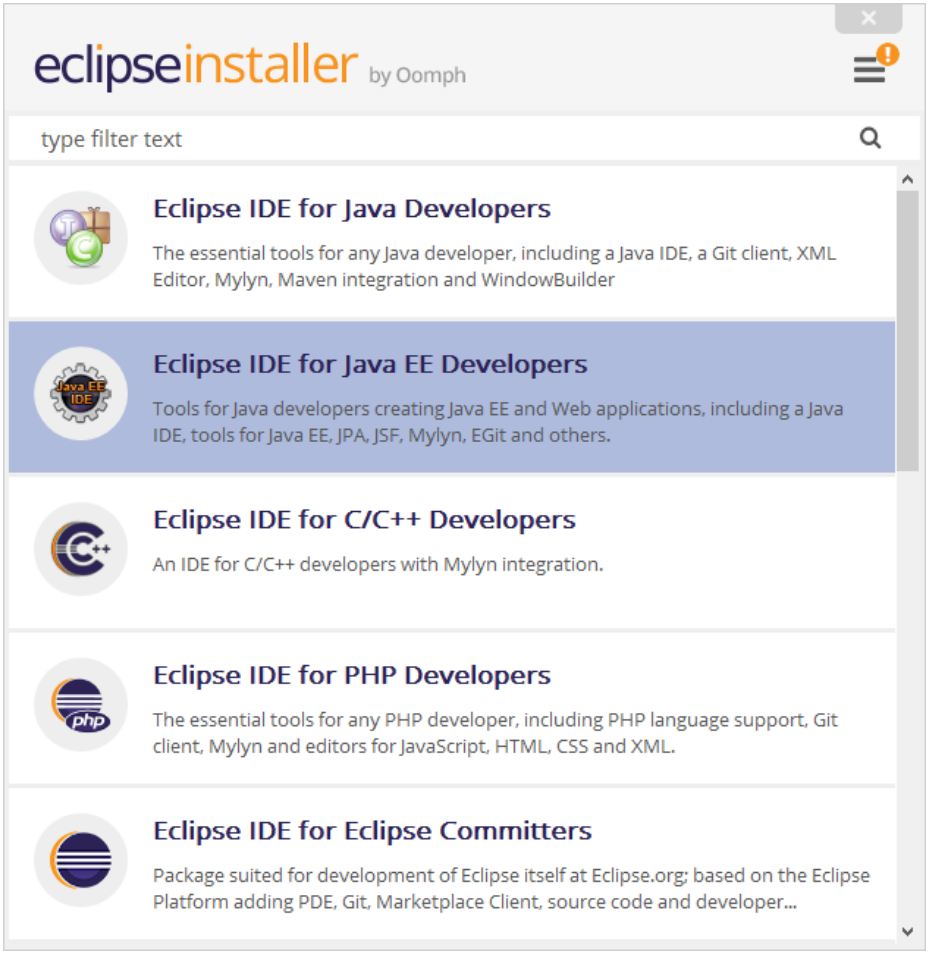


Рис. 1. Установка Eclipse

После выбора продолжить установку со всеми параметрами по умолчанию.

Сборка Java EE содержит ряд плагинов WTP (Web-Tools Platform), предназначенных для разработки Java EE компонентов: EJB, JSP, Java-сервлеты и другие. Тем не менее, необходимо понимать, что набор плагинов WTP может быть подключен к сборке Eclipse вручную (см. http://www.eclipse.org/webtools/).

Установка Wildfly

Дистрибутив сервера приложений доступен по адресу: http://wildfly.org/downloads/. В данном лабораторном практикуме была использована версия 10.0.0.

Дистрибутив представляет собой .zip файл. Для его установки следует произвести следующие действия:

1. В корне диска C: создать директорию WebDevel

2. Распаковать в эту директорию архив.
3. Убедиться, что установлена переменные среды JAVA_HOME и JBOSS_HOME. Переменные должны соответствовать путям, в которые были установлены JDK и WildFly:

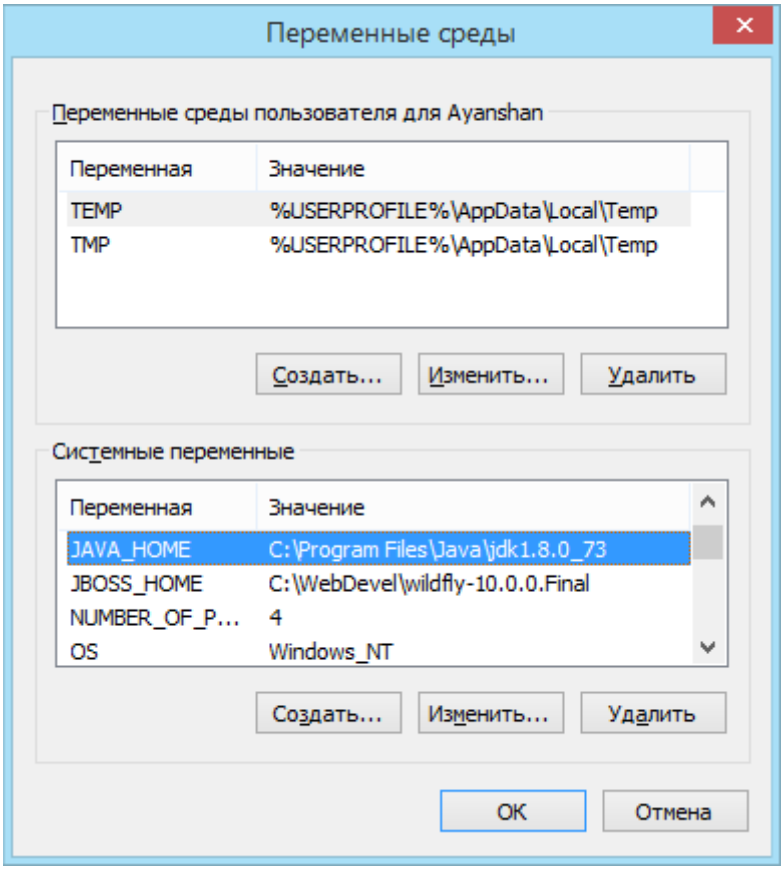


Рис. 2. Переменные среды Java и JBOSS

Для проверки работоспособности WildFly требуется запустить файл standalone.bat из директории bin установленного WildFly. Если сервер был установлен корректно, произойдет его запуск без сообщений об ошибках.

Установка JBoss Tools

Для установки JBoss Tools необходимо выполнить следующие действия:

1. Запустить Eclipse
2. В меню Help выбрать пункт Eclipse Marketplace
3. В поиске ввести JBoss или JBoss Tools, выбрать JBoss и нажать Install

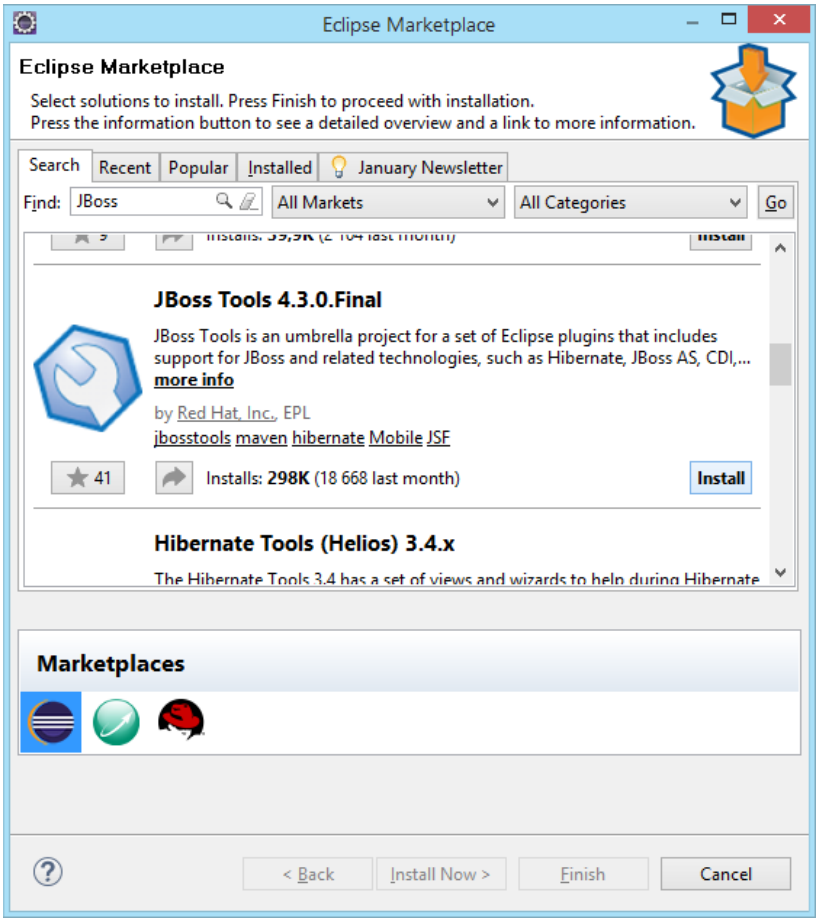


Рис. 3. Установка JBoss Tools

4. Убедиться, что все галочки отмечены и нажать кнопку Confirm

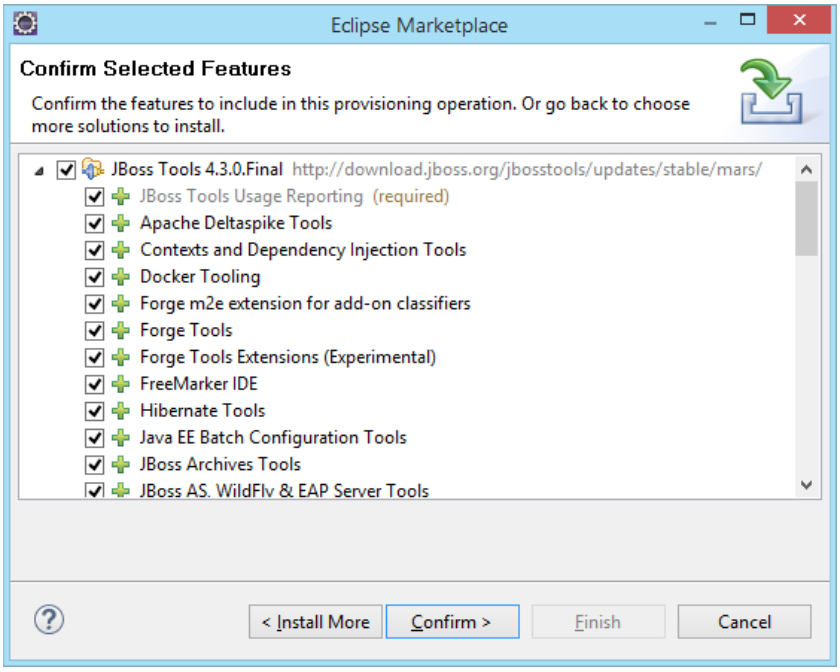


Рис. 4. Выбор компонент JBoss Tools

После этого согласиться с лицензионным соглашением и дождаться окончания установки.

5. После этого в Eclipse нужно добавить новый сервер. Выбираем File/New/Other… и в открывшемся окне выбрать Server.

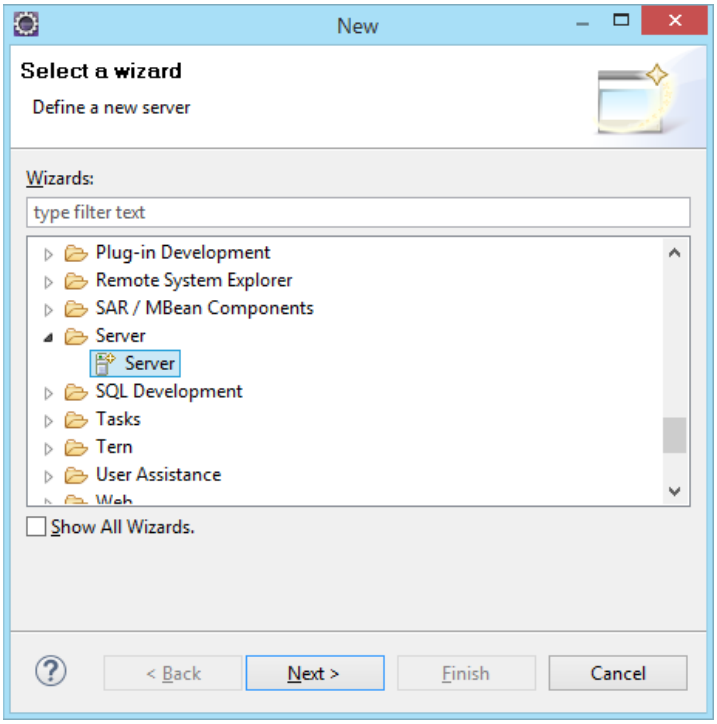


Рис. 5. Добавление сервера в Eclipse

6. После этого нажать кнопку Next и выбрать установленную версию WildFly:

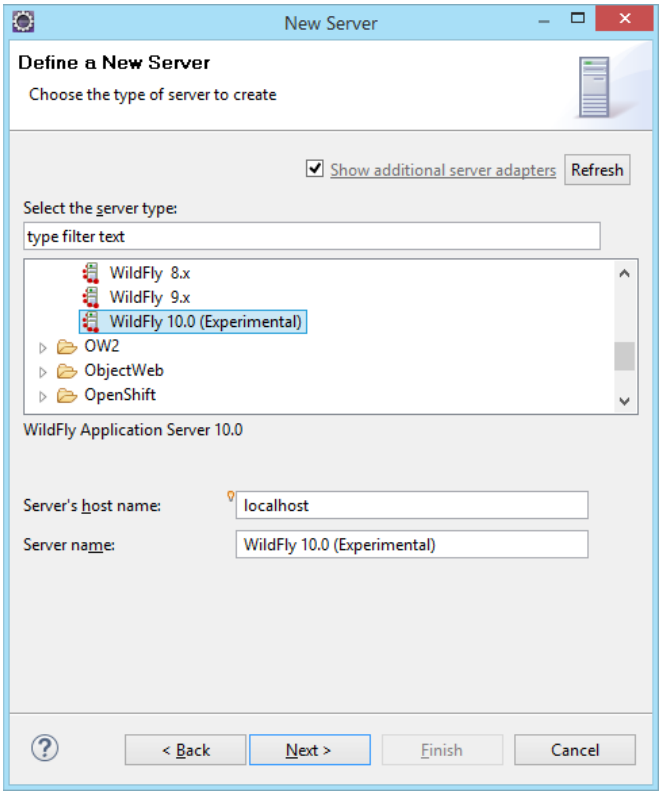


Рис. 6. Выбор сервера

7. После нажатия кнопки Next откроется окно настройки сервера. В том случае, если сервер установлен на локальной машине, все пункты можно оставить без изменений.

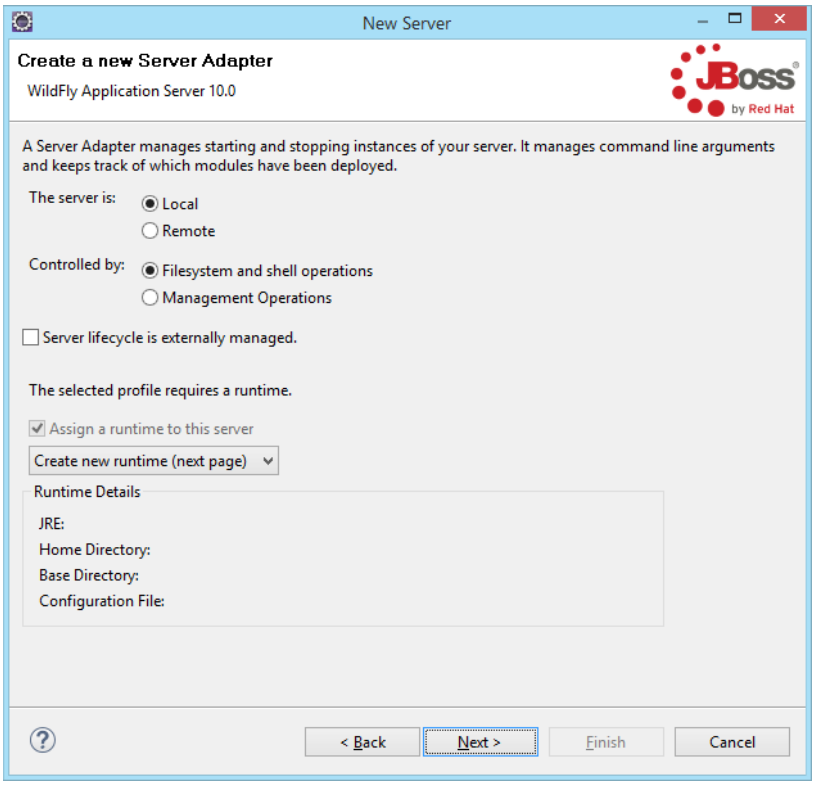


Рис. 7. Параметры сервера

8. После нажатия Next прописать путь к установленному WildFly и указать установленный JDK, затем нажать Finish:

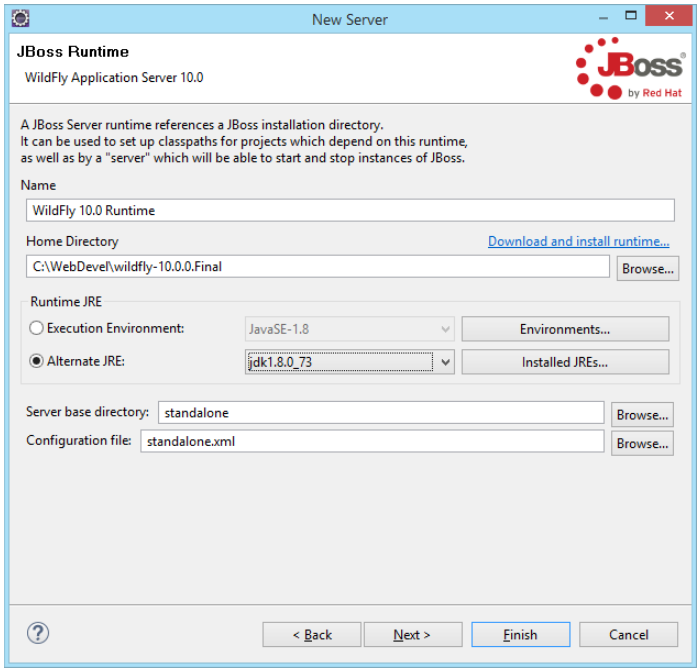


Рис. 8. Настройка путей и JRE

Установка PostgreSQL

Графический инсталлятор PostgreSQL можно скачать с сайта http://www.enterprisedb.com/products-services-training/pgdownload#windows/.

Устанавливать его следует со всеми параметрами по умолчанию, установив и запомнив пароль пользователя. После установки инсталлятор предложит запустить StackBuilder для установки дополнительных компонентов. Выбрать в нем pgAdmin и jdbc драйвер:

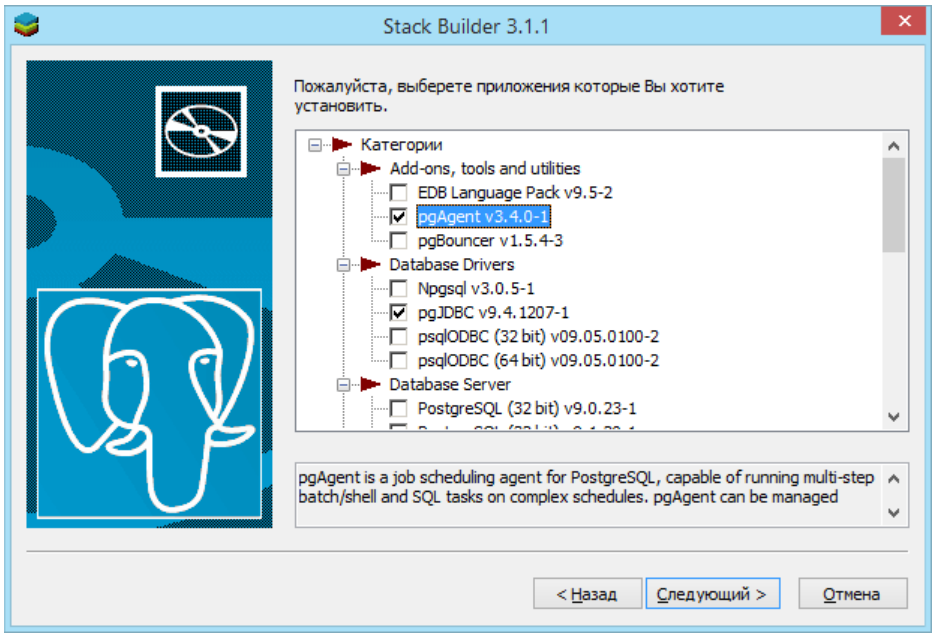


Рис. 9. Дополнительные компоненты PostgreSQL

Замечания:

1. Все перечисленное программное обеспечение является свободно распространяемым.

2. Материалы и примеры в предлагаемом курсе выполнены на базе ОС Windows, однако перечисленное программное обеспечение и лабораторные работы могут быть установлены и выполнены под управлением других операционных систем (Linux, MacOS и др.). Для установки JDK и Eclipse в операционную систему, альтернативную Windows, необходимо использовать соответствующий этой ОС дистрибутив.

Лабораторная работа №2. Создание базы данных

Цель работы: создать структуру базы данных для хранения информации о сотрудниках. В качестве СУБД должна использоваться PostgreSQL.

Выполнение работы:

1. Запустить приложение pgAdmin
2. Выбрать установленный сервер, щелкнуть правой клавишей и выбрать «Подключение»

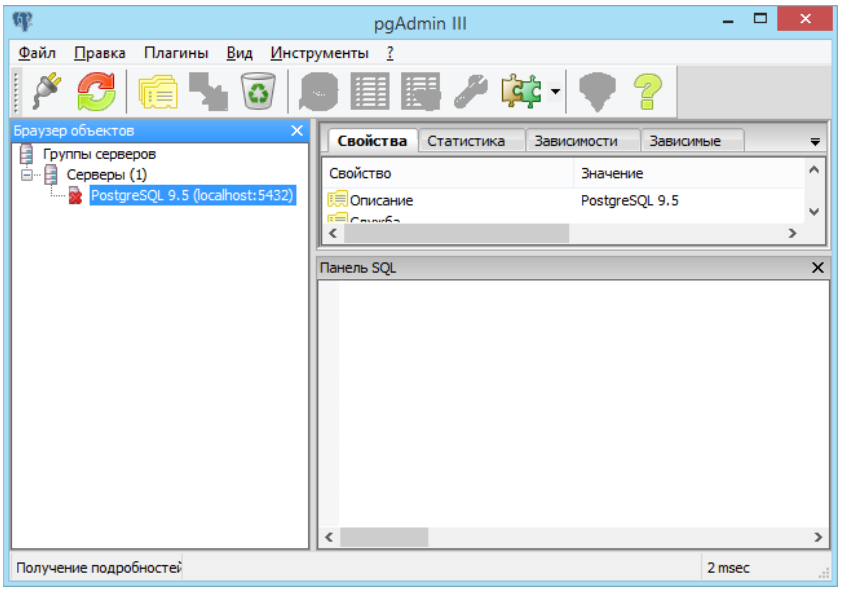


Рис. 10. Подключение к серверу

3. Ввести указанный при установке пароль суперпользователя:

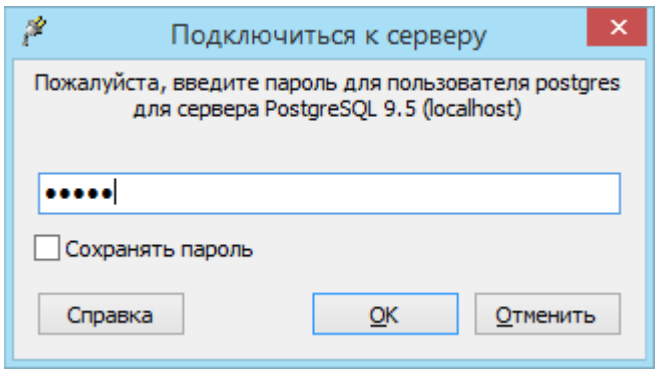


Рис. 11. Ввод пароля

4. В левой части окна выбрать «Базы данных», щелкнуть правой клавишей мыши по этому пункту и выбрать «Новая база данных».

5. В появившемся окне указать имя базы данных: EmployeeDB и нажать ОК.

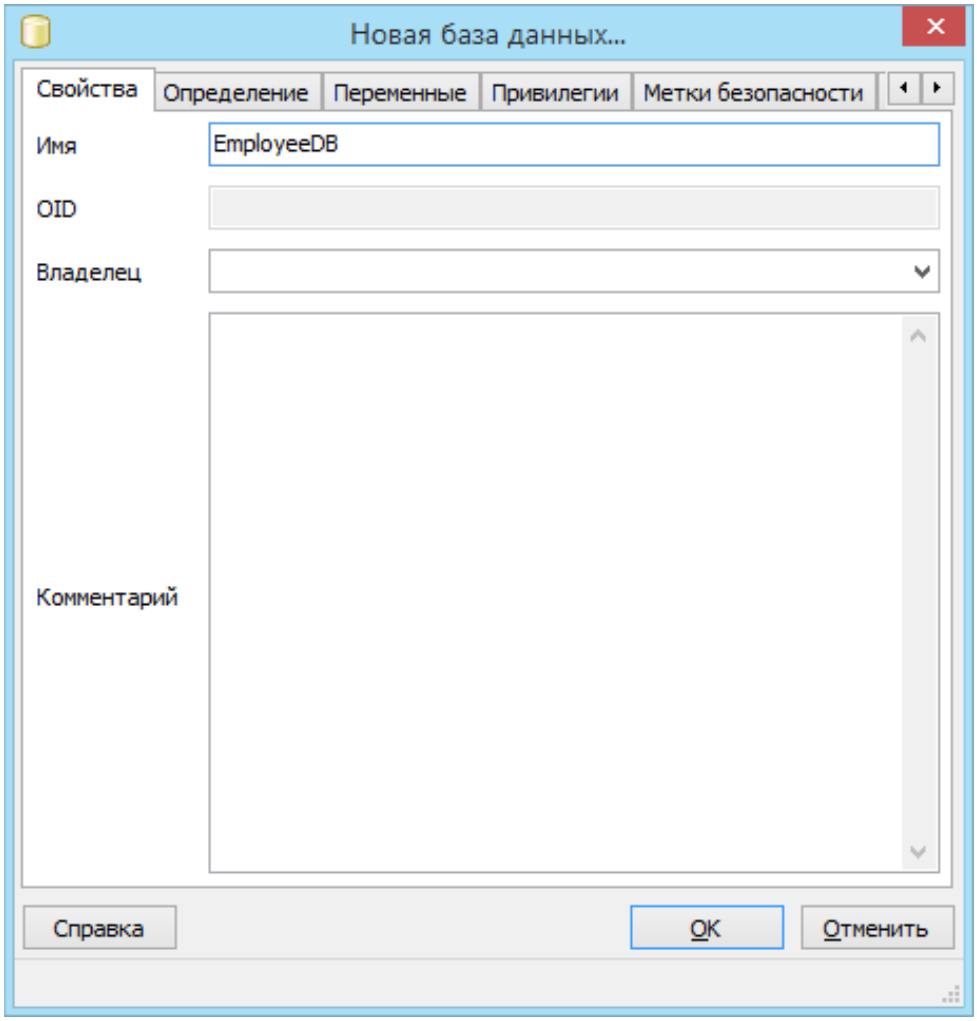


Рис. 12. Создание новой базы данных

6. Выбрать в панели инструментов окна кнопку «Пользовательские SQL-запросы»

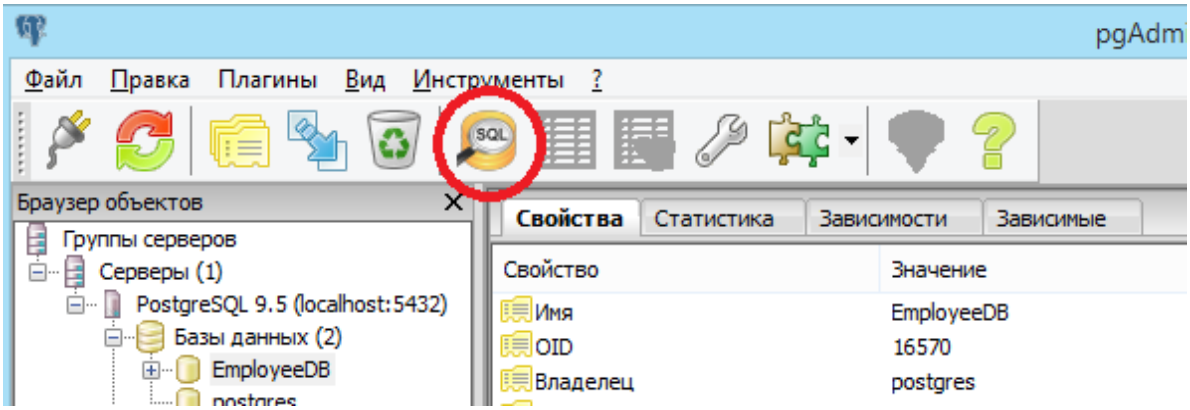
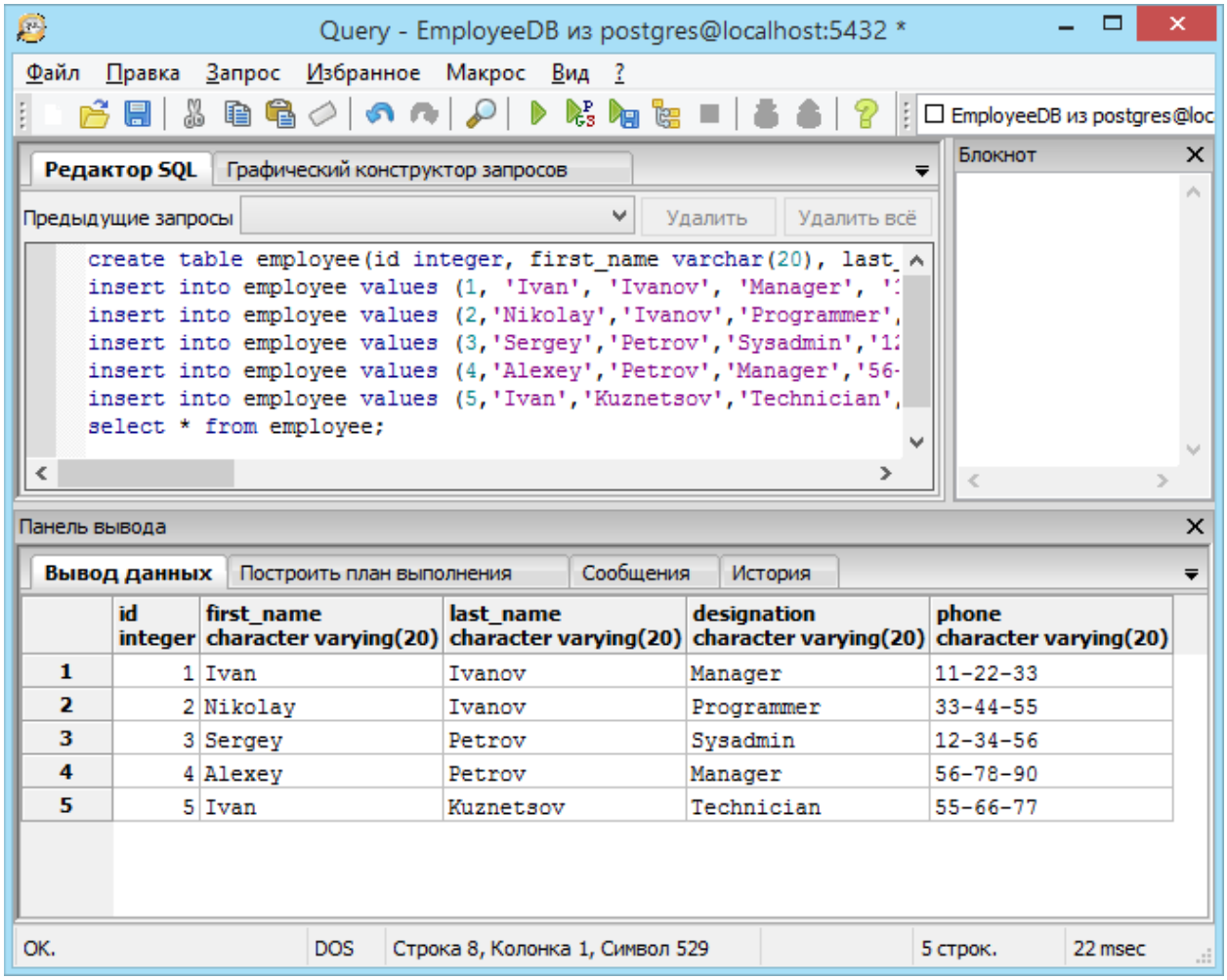


Рис. 13. Пользовательские SQL-запросы

7. В появившемся окне запроса выполнить следующий скрипт, создающий таблицу и наполняющий её данными:

```
create table employee(id integer, first_name varchar(20), last_name
varchar(20), designation varchar(20), phone varchar(20));
insert into employee values (1, 'Ivan', 'Ivanov', 'Manager', '11-22-33');
insert into employee values (2,'Nikolay','Ivanov','Programmer','33-44-55');
insert into employee values (3,'Sergey','Petrov','Sysadmin','12-34-56');
insert into employee values (4,'Alexey','Petrov','Manager','56-78-90');
insert into employee values (5,'Ivan','Kuznetsov','Technician','55-66-77');
select * from employee;
```

В случае успешного выполнения скрипта в панели вывода отобразится результат выполнения запроса SELECT - таблица с данными:



	id integer	first_name character varying(20)	last_name character varying(20)	designation character varying(20)	phone character varying(20)
1	1	Ivan	Ivanov	Manager	11-22-33
2	2	Nikolay	Ivanov	Programmer	33-44-55
3	3	Sergey	Petrov	Sysadmin	12-34-56
4	4	Alexey	Petrov	Manager	56-78-90
5	5	Ivan	Kuznetsov	Technician	55-66-77

Рис. 14. Результат выполнения запросов

Лабораторная работа №3. Создание Java Servlet

Цель работы: разработать веб-приложение, использующее сервлет для поиска информации о сотрудниках организации. Данные о сотрудниках хранятся в таблице Employee. Для осуществления поиска пользователь указывает фамилию сотрудника и просматривает информацию о найденных сотрудниках (возможно существование нескольких сотрудников с одинаковыми фамилиями).

Выполнение работы:

1. Создание нового проекта в Eclipse

   В меню File выбрать New/Other, в открывшемся окне – Web/Dynamic Web Project.

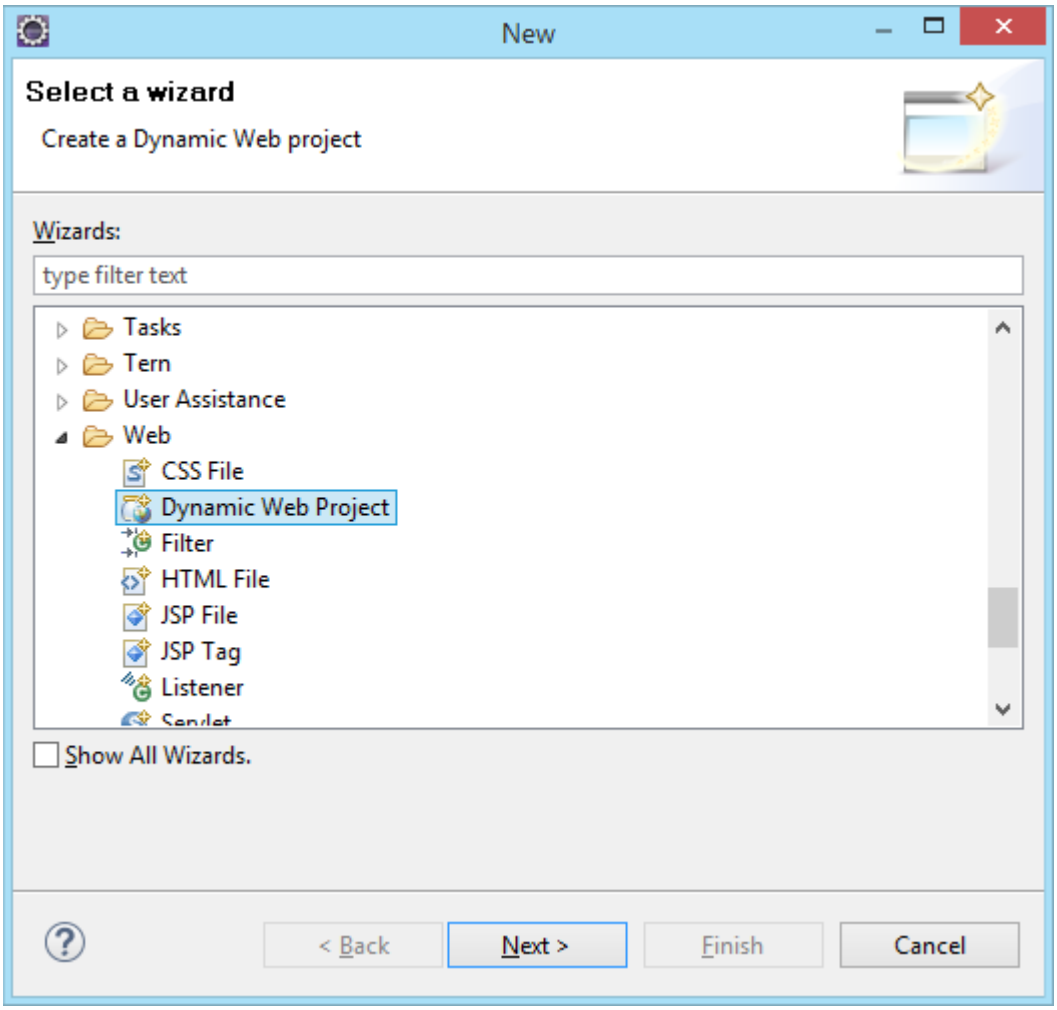


Рис. 15. Создание проекта сервлета

2. Установить имя проекта ServletLab и выбрать в качестве TargetRuntime установленный нам WildFly.

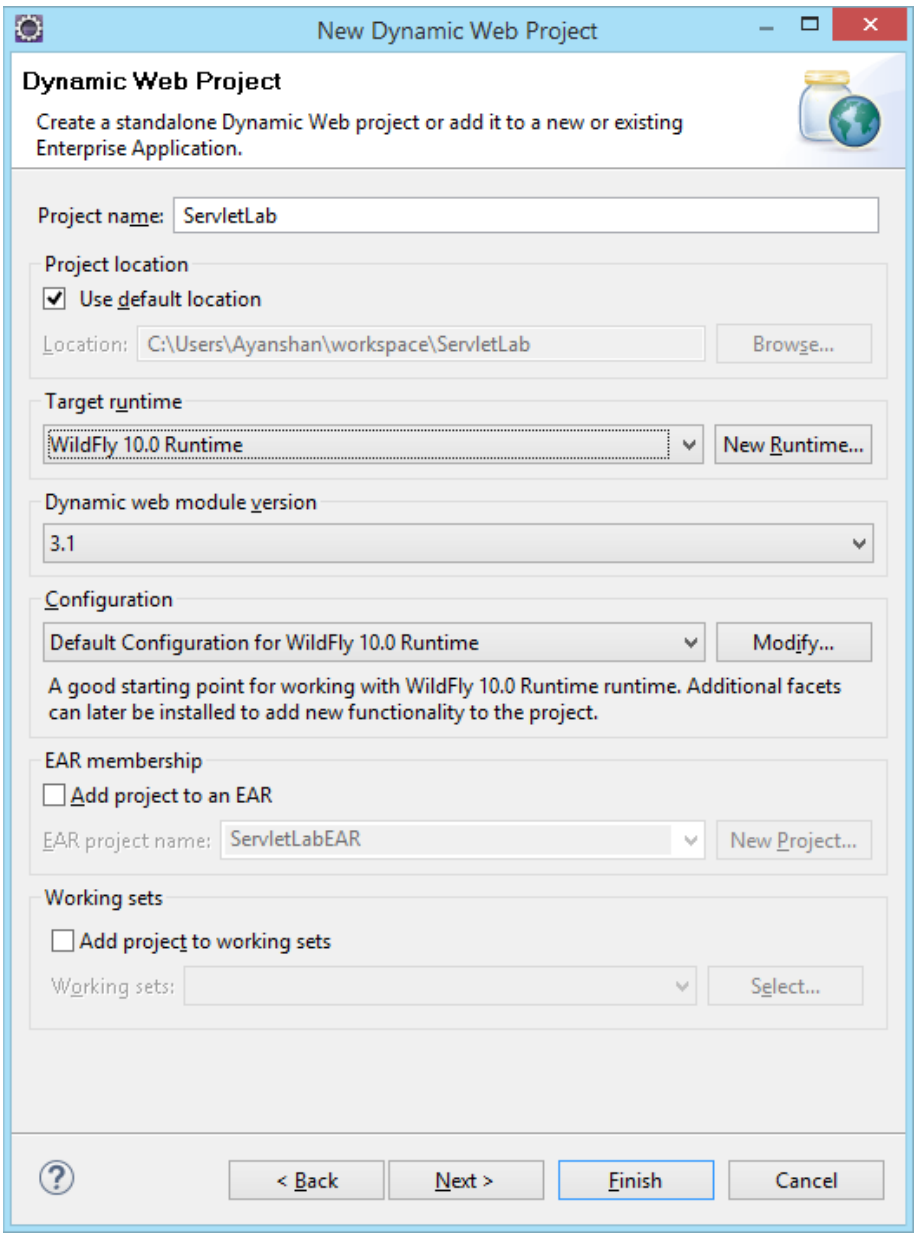


Рис. 16. Параметры проекта

Нажатие кнопки Finish приводит к созданию проекта, который отобразится в Project Explorer.

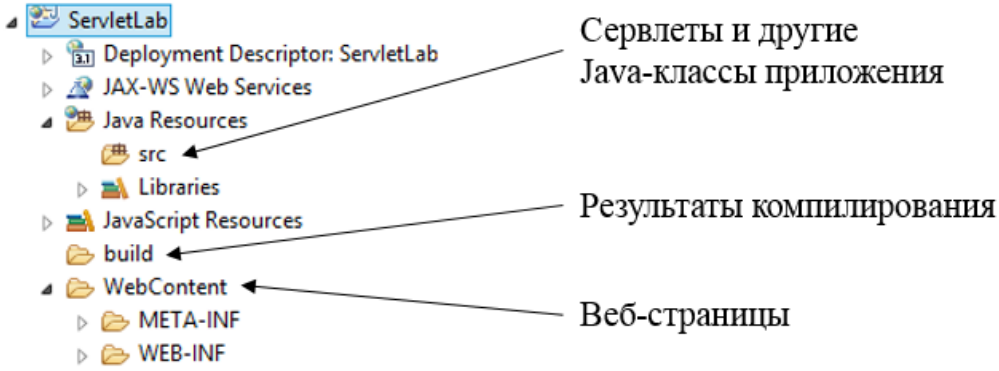


Рис. 17. Дерево проекта

Перед тем как начать программировать сервлет, создадим обычный Java-класс Employee, который будет использоваться для представления и передачи данных о сотруднике.

3. Создать новый Java-класс, нажав правой кнопкой мыши на проект ServletLab и выбрав пункт меню New/Class. Назовите класс Employee и разместите его в пакете ru.ulstu.vt.servletLab
4. В классе Employee создайте пять полей, соответствующих столбцам таблицы employee, добавьте конструкторы и набор get/set методов. Полный код класса Employee приведен ниже:

Прежде чем приступить к разработке сервлета, отключите пункт Project/Build Automatically в главном меню Eclipse. Это позволит избежать автоматической развертки приложения на сервер при сохранении исходных файлов.

Теперь можно переходить к созданию сервлета.

5. Для создания нового сервлета щелкните правой кнопкой мыши на проект ServletLab, выберите пункт меню New/Other, укажите Web/Servlet и нажмите Next.
6. Укажите пакет (Java package), в котором будет размещен сервлет, например, ru.ulstu.vt.servletLab.
7. Укажите имя класса сервлета (Class name) EmployeeServlet и нажмите Finish

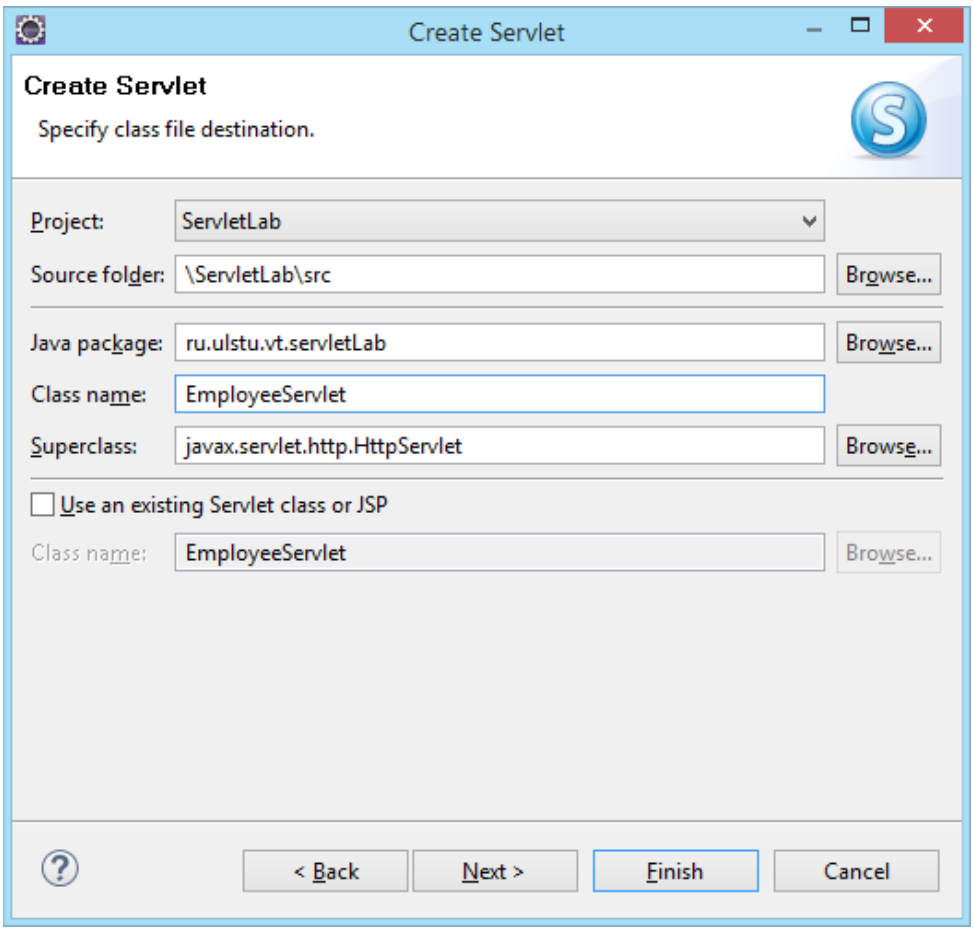


Рис. 18. Создание сервлета

8. Программный код сервлета приведен ниже. Основная бизнес-логика сосредоточена в методе doGet(), который выполняется на сервере после того, как пользователь запустил сервлет на выполнение. Скопируйте код и сохраните сервлет, нажав Ctrl-S.

```
package ru.ulstu.vt.servletLab;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class EmployeeServlet
 */
@WebServlet("/EmployeeServlet")
public class EmployeeServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
```

```
	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public EmployeeServlet() {
		super();
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse
response)
			throws ServletException, IOException {
		// TODO Auto-generated method stub
		try {
			response.setContentType("text/html;charset=UTF-8");
			// Получение из http-запроса значения параметра lasname
			String lastname = request.getParameter("lastname");

			// Коллекция для хранения найденных сотрудников
			ArrayList<Employee> employees = new ArrayList<Employee>();

			// Загрузка драйвера БД PostgreSQL
			Class.forName("org.postgresql.Driver").newInstance();

			// Получение соединения с БД
			Connection con =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/EmployeeDB",
"postgres",
					"admin");
			// Выполнение SQL-запроса
			ResultSet rs = con.createStatement().executeQuery("Select
id, first_name, last_name, designation, phone "
					+ "From employee " + "Where last_name like '" +
lastname + "'");
			// Перечисление результатов запроса
			while (rs.next()) {
				// По каждой записи выборки формируется
				// объект класса Employee.
				// Значения свойств заполняются из полей записи
				Employee emp = new Employee(rs.getLong(1),
rs.getString(2), rs.getString(3), rs.getString(4),
						rs.getString(5));
				// Добавление созданного объекта в коллекцию
				employees.add(emp);
			}
			// Закрываем выборку и соединение с БД
			rs.close();
			con.close();

			// Выводим информацию о найденных сотрудниках
			PrintWriter out = response.getWriter();
			out.println("Найденные сотрудники<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);
            }

        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
         *      response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse
response)
                throws ServletException, IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
        }

}
```

9. В том случае, если в папке WebContent/WebInf проекта отсутствует файл web.xml, требуется щелкнуть правой клавишей мыши по имени проекта в Projects Explorer и выбрать пункт Java EE Tools/Generate Deployment Descriptor Stub.
10. Добавить информацию о сервлете в файл Web.xml:

10.1. Дважды щёлкнуть мышью по файлу Web.xml. Откроется Web XML Editor:

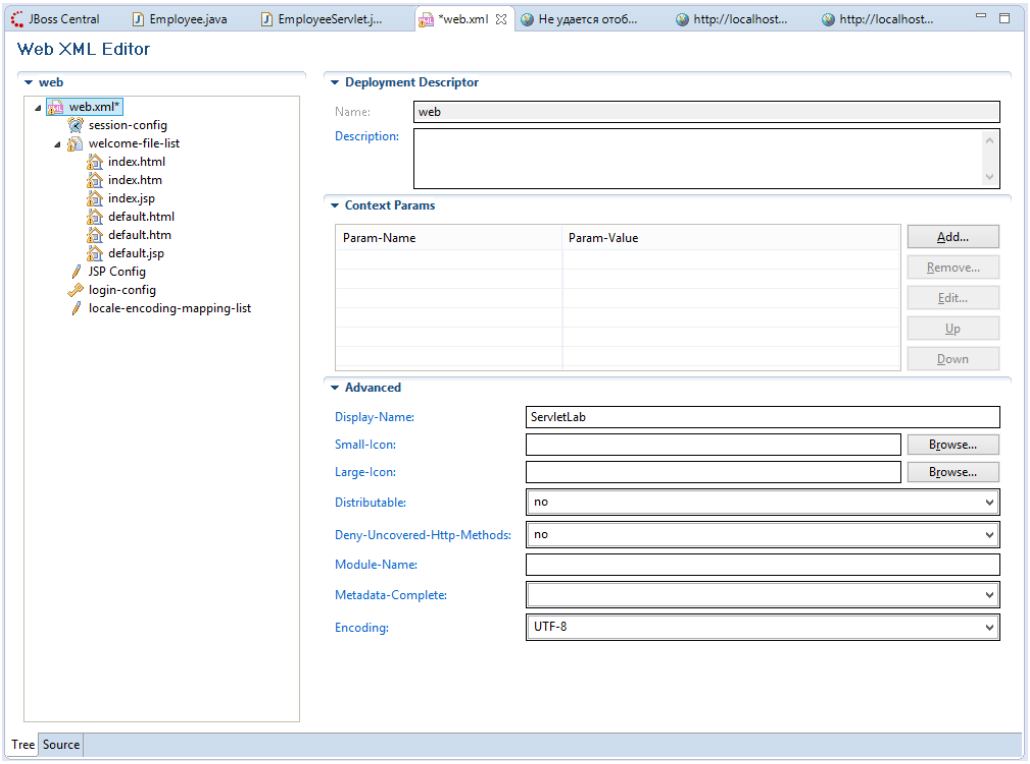


Рис. 19. Web XML редактор

10.2. В левой части окна щёлкнуть правой клавишей мыши на web.xml и выбрать New/Servlet… В появившемся окне указать информацию о создаваемом сервлете и нажать Finish

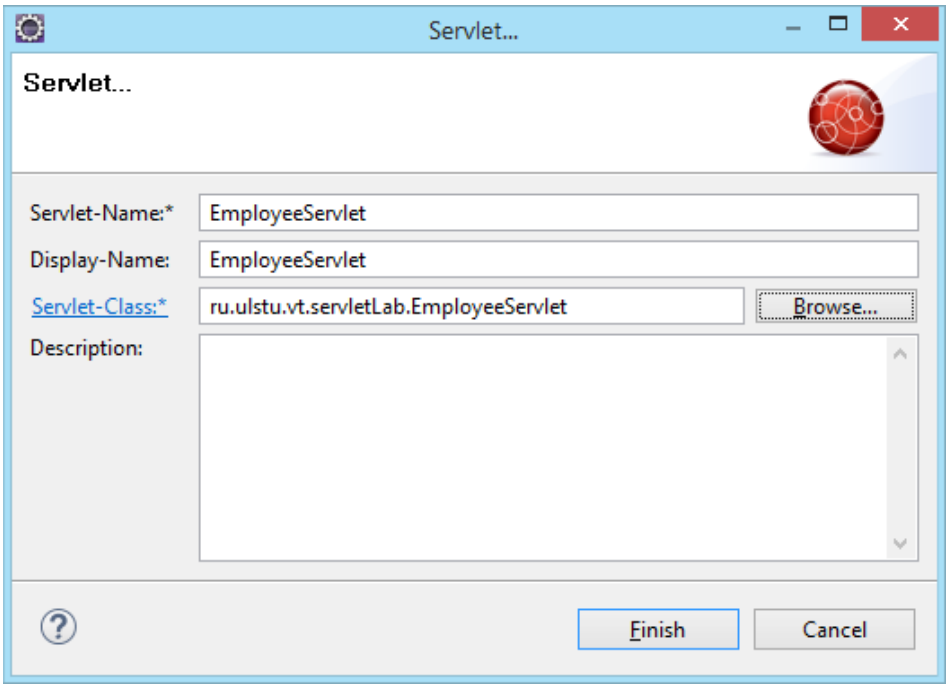


Рис. 20. Описание сервлета

10.3. Опять левой клавишей мыши щелкнуть web.xml. Выбрать New/Servlet Mapping… В появившемся окне указать:

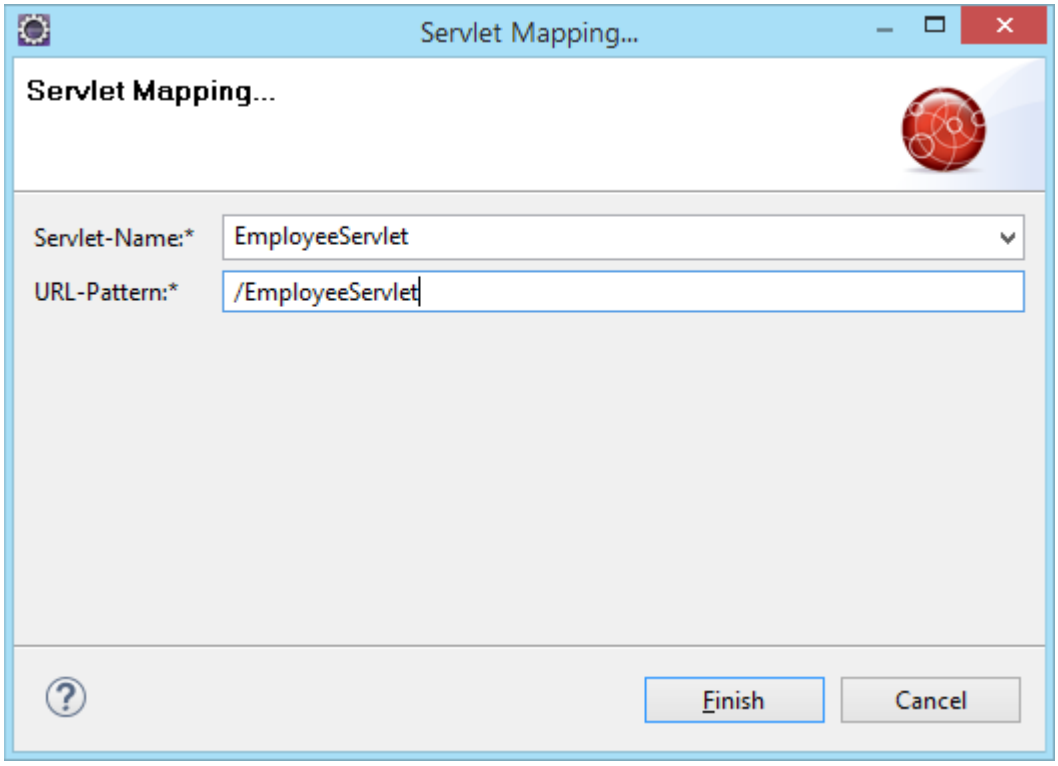


Рис. 21. Описание размещения сервлета

10.4. Дерево Web XML Editor должно принять следующий вид:

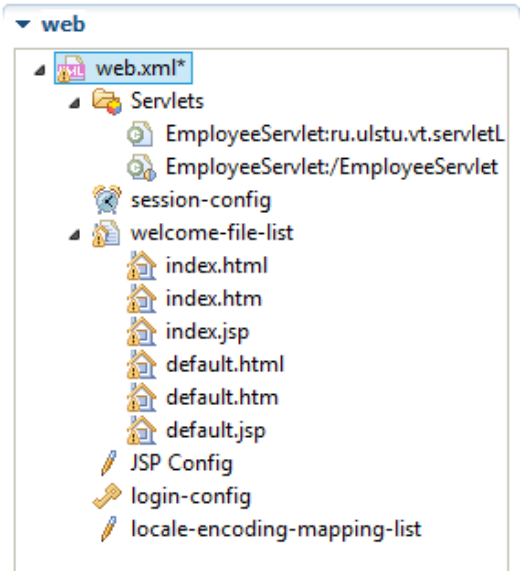


Рис. 22. web.xml с описанием сервлета

11. В проект добавить jdbc драйвер PostgreSQL. Для этого найти директорию, в которую он был установлен в процессе установки PostgreSQL, и перетащить его мышью в папку WebContent/WEB-INF/lib проекта:

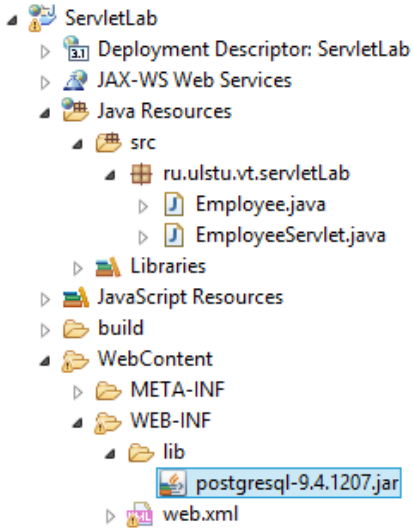


Рис. 23. JDBC-драйвер

12. Теперь всё готово для запуска сервлета на сервере. Для этого нужно правой клавишей мыши щелкнуть по имени проекта и выбрать Build Project. В случае успешного построения проекта в том же меню выбрать Run as/Run on Server и выбрать установленный сервер WildFly в открывшемся окне. При этом открывается вкладка

браузера, а в нижней части окна – консольный вывод сервера. В случае, если он установлен и настроен корректно, окно будет выглядеть следующим образом:

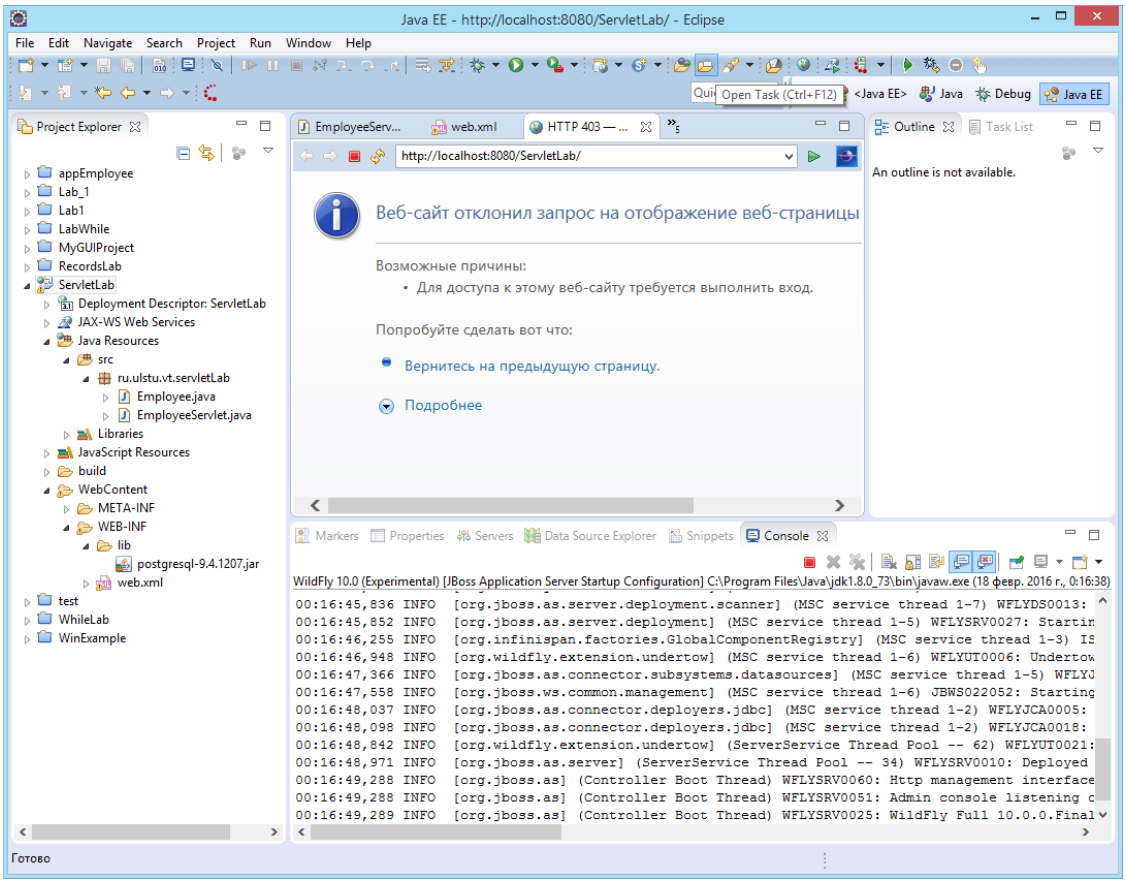


Рис. 24.Запуск сервлета на сервере

13. На вкладке браузера или в отдельно запущенном браузере перейти по ссылке http://localhost:8080/ServletLab/EmployeeServlet?lastname=Ivanov
14. В том случае, если всё было выполнено успешно, отобразится окно следующего содержания:

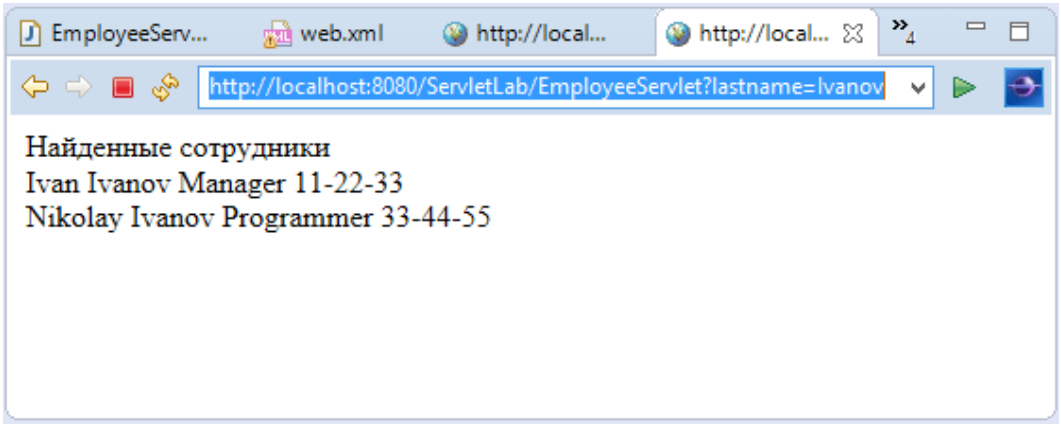


Рис. 25. Результат обращения к сервлету

Лабораторная работа №4. Создание JSP-страницы

Цель работы: доработать веб-приложение, организовав веб-интерфейс приложения, в котором пользователь указывает фамилию сотрудника, выполняет поиск и просматривает информацию о найденных сотрудниках. Интерфейс требуется реализовать в виде JSP-страницы.

Выполнение работы: Для решения поставленной задачи необходимо выполнить следующие шаги:

- Разработать JSP страницу для ввода исходных данных и отсылки сообщения сервлету.
- Доработать код сервлета для отправки результатов поиска jsp-странице.
- Упаковать приложение и развернуть на сервере.
- Протестировать работу приложения в браузере.

Создание индексной страницы

Наш проект будет содержать всего одну страницу index.jsp, которая будет использоваться и для ввода фамилии, и для отображения результатов поиска.

1. Для создания новой JSP-страницы нужно нажать правой кнопкой мыши на проект ServletLab и выбрать пункт меню New/Other, затем выбрать Web/JSP File и нажать Next.
2. Указать имя файла index.jsp и убедиться, что в дереве структуры проекта выбран каталог WebContent. Нажать Finish.

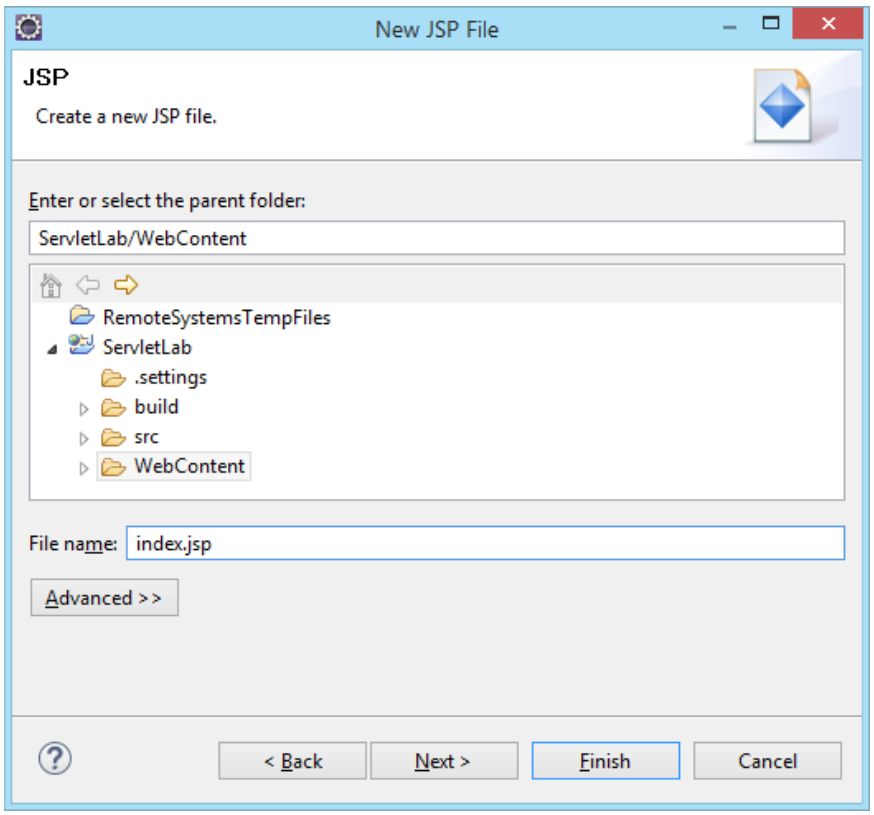


Рис. 26. Создание JSP-страницы

3. Созданный JSP-файл уже содержит базовый набор тэгов заголовка и тела веб-страницы. Нам необходимо добавить следующие элементы кода:

- поменять кодировку страницы на UTF-8 для корректного отображения символов кириллицы

- поменять заголовок страницы на «Поиск сотрудников»

- создать форму, включающую в себя текстовое поле lastname для ввода фамилии и кнопку подтверждения (Submit). При выполнении подтверждения форма передает значение параметра lastname сервлету EmployeeServlet.

- сигналом того, что поиск был выполнен и имеются результаты для отображения, будет являтся наличие параметра http-запроса employeesFound, который будет передан сервлетом странице index.jsp в случае успешного выполнения поиска. Далее, проверив размер коллекции найденных сотрудников, мы определим, был ли найден хотя бы один сотрудник. В случае утвердительного ответа обработаем коллекцию, и отобразим все свойства каждого найденного сотрудника в виде таблицы.

Ниже приведен полный код JSP-страницы:

```
<%@ 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="ru.ulstu.vt.servletLab.Employee"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Поиск сотрудников</title>
</head>
<body>
      <form action="EmployeeServlet">
            Фамилия сотрудника
            <input type="text" name="lastname">
            <input type=submit value="поиск">
      </form>
      <%
      // Получение значения параметра employeesFound
      ArrayList employees = (ArrayList)
      request.getAttribute("employeesFound");
      // Если параметр задан, начинаем обработку
      if (employees != null) {
            // Если не найдено ни одного сотрудника - вывод сообщения
            if (employees.size()==0)
                  out.print("Сотрудники не найдены");
            else {
            out.print("<TABLE border=\"1\">");
            // Заголовок таблицы
            out.print("<TR><TD>Id</TD><TD>Имя</TD><TD>Фамилия</TD>" +
                        "<TD>Должность</TD><TD>Телефон</TD></TR>");
            for (int i = 0; i < employees.size(); i++) {
                  // По каждому найденному сотруднику
                  // формируется строка таблицы
                  out.print("<TR>");
                  // Получение очередного сотрудника из коллекции
                  Employee emp = (Employee) employees.get(i);
                  // Заполнение строки таблицы свойствами сотрудника
                  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>
```

Доработка сервлета

Метод сервлета doGet() выполняется после того как пользователь выполнил подтверждение (Submit) формы. Необходимо изменить код сервлета таким образом, чтобы результаты поиска отправлялись jsp-странице в виде параметра employeesFound. Для этого закомментируем часть кода, связанную с выводом коллекции сотрудников в поток вывода (response.getWriter()), и

добавим код помещения этой коллекции в параметр запроса и перенаправление запроса к странице index.jsp:

```
/*
// Выводим информацию о найденных сотрудниках
PrintWriter out = response.getWriter();
out.println("Найденные сотрудники<br>");
for (Employee emp: employees) {
      out.print(emp.getFirstName() + " " +
                  emp.getLastName() + " " +
                  emp.getDesignation() + " " +
                  emp.getPhone() + "<br>");
}
*/
// Помещение результатов в параметр запроса employeesFound
request.setAttribute("employeesFound", employees);
// Перенаправление http-запроса на страницу index.jsp
RequestDispatcher dispatcher = getServletContext()
            .getRequestDispatcher("/index.jsp");
dispatcher.forward(request, response);
```

Также потребуется подключить библиотеку RequestDispatcher:

```
import javax.servlet.RequestDispatcher;
```

Тестирование приложения

Для тестирования разработанного приложения требуется запустить его на сервере. Для этого нужно сначала построить проект, а затем выбрать ServletLab в Projects Explorer и щёлкнуть его правой клавишей мыши. После этого выбрать Run As/Run on Server… и запустить его на установленном сервере WildFly. После этого запустить браузер и открыть страницу по адресу:

http://localhost:8080/ServletLab/index.jsp

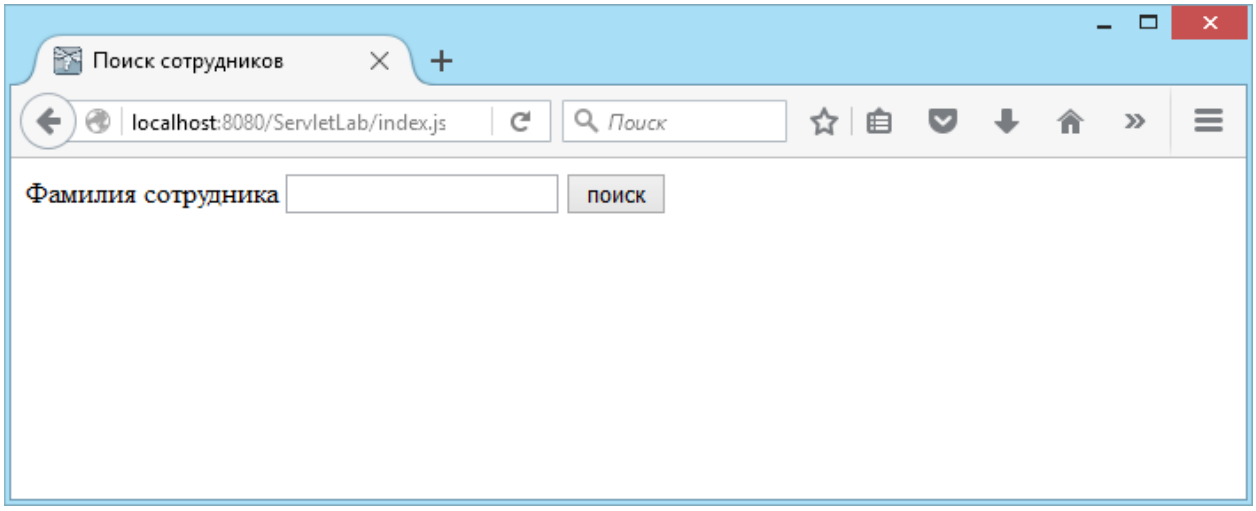


Рис. 26. Выполнение Web-приложение

После ввода фамилии Ivanov и нажатия кнопки «поиск» должна отобразиться таблица результатов:

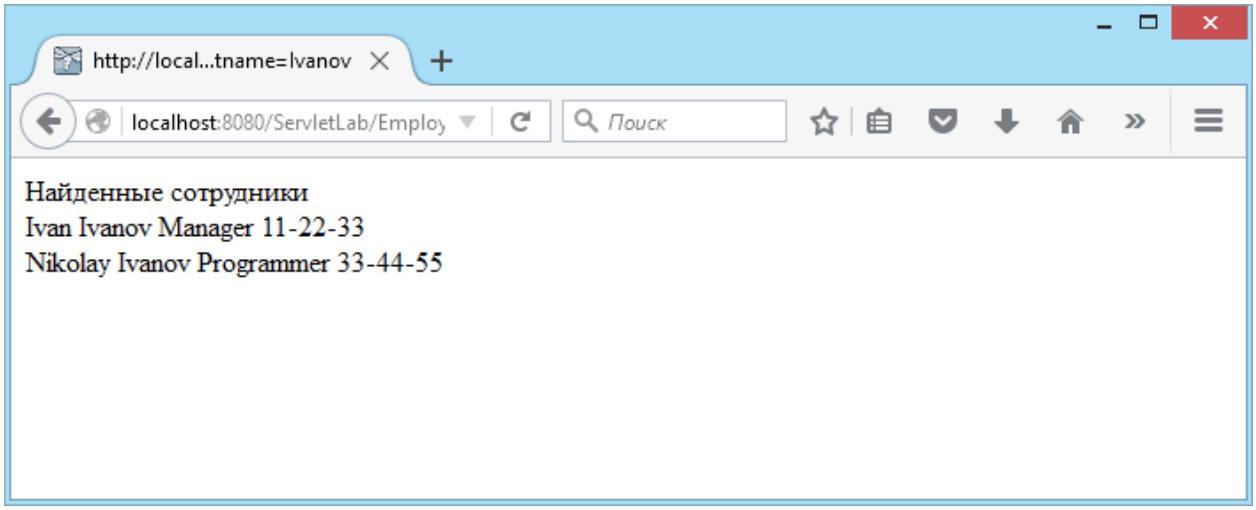


Рис. 27. Результат поиска