

ЛАБОРАТОРНАЯ РАБОТА 3

Тема: Корпоративные технологии Java EE. Технология сервлетов

Сервлетами называют небольшие программы, которые выполняются на стороне сервера веб-подключения. Сервлеты динамически расширяют функциональные возможности веб-сервера.

Обработка запроса статической веб-страницы происходит в следующей последовательности: в браузере пользователь вводит в поле адреса URL (Uniform Resource Locator - унифицированный указатель информационного ресурса), браузер формирует запрос по протоколу HTTP, направляя его соответствующему веб-серверу, который сопоставляет этот запрос с конкретным файлом, возвращая его браузеру в виде ответа через протокол HTTP. HTTP-заголовок ответа обозначает тип содержимого. Если тип MIME (Multipurpose Internet Mail Extensions) обозначен как text/plain, то это обычный текст в формате ASCII, а если — text/html, то это HTML-документ.

Если содержимое веб-страницы должно зависеть от времени получения запроса к ней, то такая страница называется динамической, т. е. ее содержимое изменяется. Как правило, веб-страница, которая формируется как ответ на запрос к базе данных, является динамической. Например, на какие-либо запросы в Интернет-магазине в ответ формируются динамические веб-страницы. Использование технологии сервлетов для формирования динамических веб-страниц имеет ряд особенностей: сервлет выполняется в адресном пространстве веб-сервера; обработка нескольких запросов клиента может выполняться в одном процессе; сервлеты не зависят от конкретной платформы, так как разрабатываются на Java; диспетчер безопасности Java на сервере накладывает ряд ограничений для защиты его ресурсов; для сервлета доступны все функциональные возможности библиотек классов Java. Сервлет может через механизмы сокетов и удаленного вызова методов (RMI) связываться с апплетами, базой данных и другим программным обеспечением.

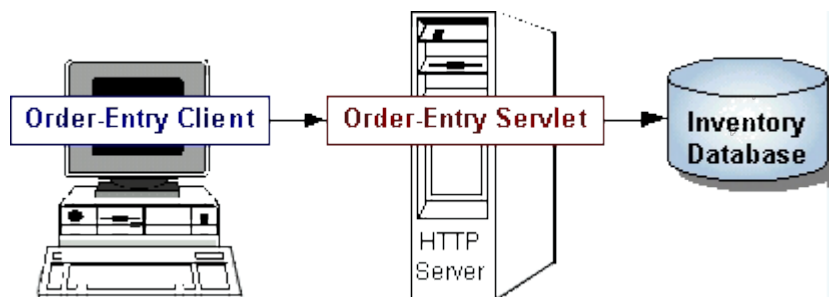


Рис.1. Использование сервлета для передачи информации из HTML-формы в базу данных

Сервлеты могут быть встроены в различные сервера. Интерфейс, который используется для написания сервлетов, не знает о среде сервера или протоколе. По-сути, сервлеты являются модулями расширения для запрос-ответ ориентированных серверов (web-сервер с поддержкой Java). Они разрабатываются на стандартном расширении Java - Java Servlet API, поэтому обходят проблему программирования серверов с платформозависимыми интерфейсами.

Основная терминология технологии сервлетов

servlet (сервлет) - Java программа, которая расширяет функциональные возможности Web-сервера, динамически генерируя содержание и взаимодействуя с Web-клиентами при помощи принципа запрос-ответ.

servlet container (контейнер сервлета) - контейнер, обеспечивающий сетевые службы, при помощи которых посылаются запросы и ответы, декодируются запросы и форматируются ответы.

Все контейнеры сервлетов должны поддерживать HTTP-протокол, но могут также поддерживать дополнительные протоколы, например, HTTPS.

servlet container, distributed (распределенный контейнер сервлета) — контейнер сервлета, запускающий Web-приложения, которые помечены как распределенные и выполняются на нескольких виртуальных машинах Java. При этом виртуальные машины могут быть запущены, как на одном, так и на разных компьютерах.

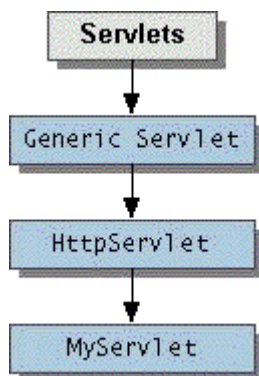
servlet context (контекст сервлета) — объект, содержащий представление (вид) Web-приложения, в котором запущен сервлет. Используя контекст, сервлет может вести журнал событий, получать URL-ссылки на ресурсы, а также устанавливать и хранить атрибуты, которые могут использоваться другими сервлетами в приложении.

servlet mapping (отображение сервлета) — определяет связь между структурой URL и сервлетом. Используется для отображения запросов в сервлеты. Если контейнер, обрабатывающий запрос, является JSP-контейнером, то неявно отображается URL, содержащий расширение .jsp.

Создание сервлетов

Сервлеты создаются как дочерние классы (классы расширения) пакета javax.servlet, которые предоставляет интерфейсы и классы для их написания. Это позволяет через классы и интерфейсы пакета javax.servlet описывать и определять контракты между классом сервлета и средой выполнения. Среда выполнения предоставляется экземпляру класса сервлета при помощи соответствующего контейнера сервлета. Полное описание пакета можно найти по ссылке <http://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html>

Главным понятием среди интерфейсов сервлета является интерфейс Servlet (<http://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html>). Все сервлеты реализуют этот интерфейс либо на прямую, либо обычным способом, наследуя класс, который реализует его, например как HttpServlet(in the API reference documentation)



Интерфейс Servlet объявляет, но не реализует методы, которые управляют сервлетом и его общением с клиентами. При написании сервлета разработчик должен реализовать некоторые или все методы интерфейса сервлет.

Для обеспечения взаимодействия с клиентом используются два объекта:

- ServletRequest, который устанавливает связь от клиента к серверу;
- ServletResponse, который устанавливает связь от сервлета обратно к клиенту.

Оба создаются на основе соответствующих интерфейсов пакета javax.servlet.

Интерфейс ServletRequest <http://docs.oracle.com/javaee/7/api/javax/servlet/ServletRequest.html> обеспечивает сервлету доступ к предоставляемым клиентом именам параметров, используемому протоколу, имени удаленного хоста, который выполнил запрос и сервера, который его получил, а также к входному потоку, ServletInputStream(in the API reference documentation). Сервлеты используют входной поток для получения данных от клиентов, которые используют протоколы уровня приложений и такие методы как HTTP POST и PUT. Интерфейсы, наследующие интерфейс ServletRequest позволяют сервлетам получать более специфичные данные протокола.

Например, интерфейс `HttpServletRequest` содержит методы для получения специальной информации HTTP заголовков.

Интерфейс `ServletResponse` предоставляет сервлету методы, для отправки сообщений клиенту (<http://docs.oracle.com/javaee/7/api/javax/servlet/ServletResponse.html>). Он позволяет сервлету установить длину содержимого и тип MIME ответа, а также устанавливает выходной поток, `ServletOutputStream`, и `Writer` через который сервлет может отправить данные ответа.

Интерфейсы, наследующие интерфейс `ServletResponse` предоставляют сервлетам более специфичные возможности. Например, интерфейс `HttpServletResponse` содержит методы позволяющие сервлету манипулировать специальной информации HTTP заголовков.

Практическая часть

Постановка задачи

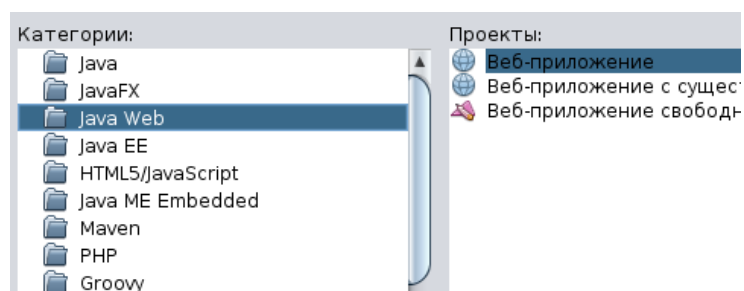
Необходимо разработать веб-приложение, использующее сервлет для поиска информации о сотрудниках организации. Данные о сотрудниках хранятся в таблице `Employee`. Для осуществления поиска пользователь указывает фамилию сотрудника и просматривает информацию о найденных сотрудниках (возможно существование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

Для решения поставленной задачи необходимо выполнить следующие шаги:

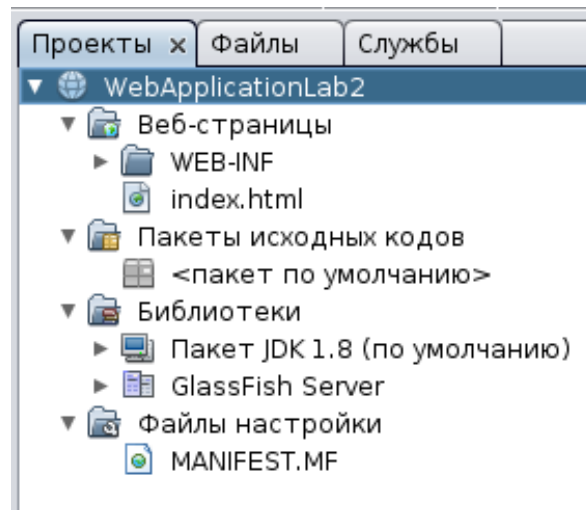
1. Создать новый проект
2. Создать таблицу `employee` и заполнить ее данными
3. Разработать сервлет, который выбирает из БД записи, соответствующие запросу пользователя и отображает результат.
4. Упаковать приложение и развернуть на сервере.
5. Протестировать работу приложения в браузере

Создание нового проекта

- 1) Выберите пункт меню Файл/Создать проект, в окне выбора типа проекта укажите Java Web/Веб-приложение и нажмите Далее.

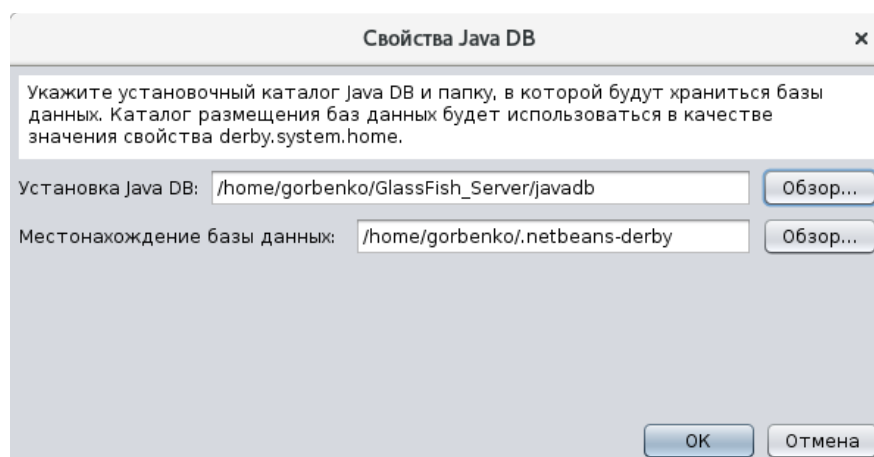


- 2) При необходимости измените имя проекта (`WebApplicationLab2`) и нажмите Далее. Если необходимо, уточните настройки сервера, на котором будет разворачиваться веб-приложение и платформу Java. Нажмите Готово.
- 3) В результате будет создан проект следующей структуры:

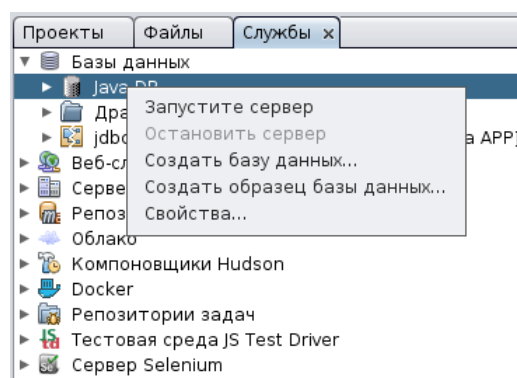


Создание таблицы employee и вставка тестовых данных

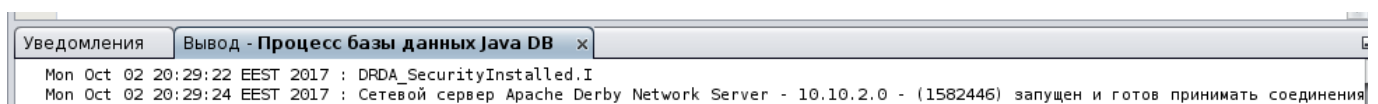
- 1) Проверяем и при необходимости изменяем установки JavaDB



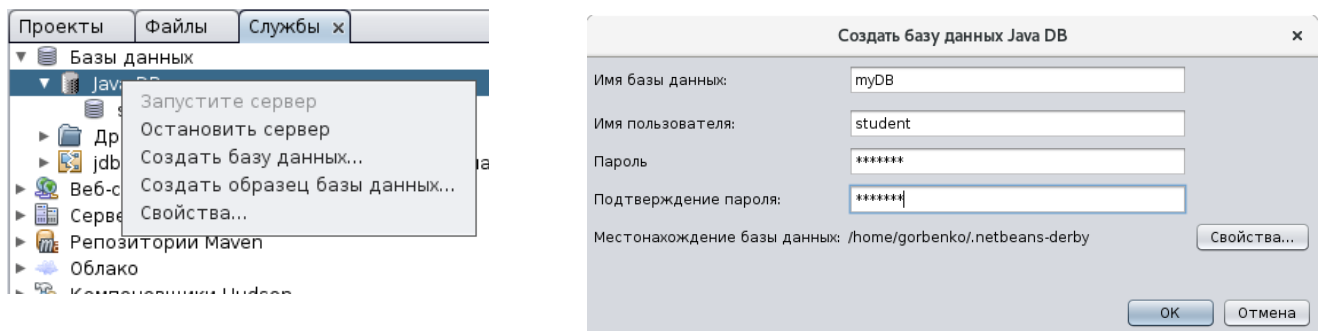
- 2) Запускаем сервер



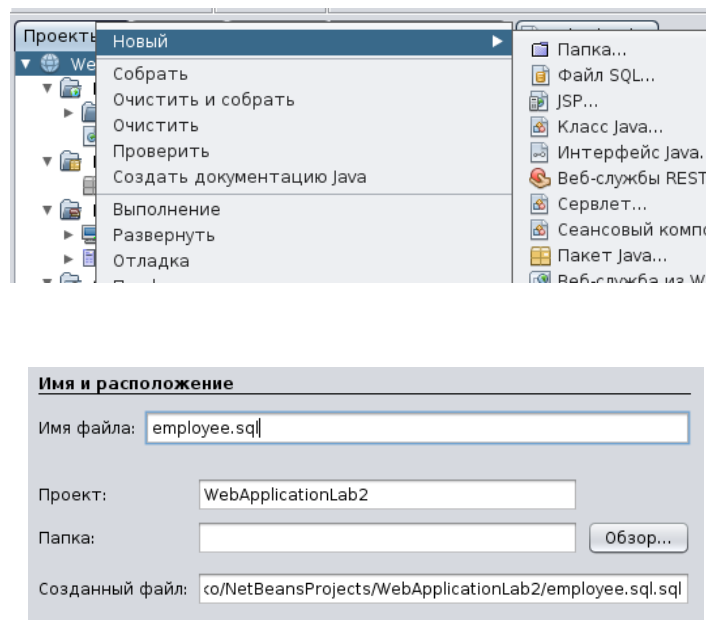
Если запуск успешный, то в консоле сообщений появится



3) Создаем новую базу данных



4) Для хранения SQL-скриптов создадим новый файл employee.sql.



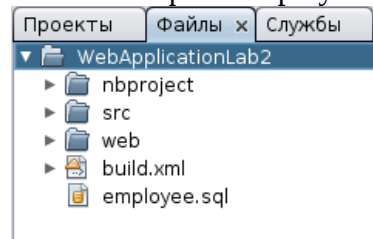
5) Созданный файл автоматически открывается для редактирования. Скопируйте в файл следующие команды:

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

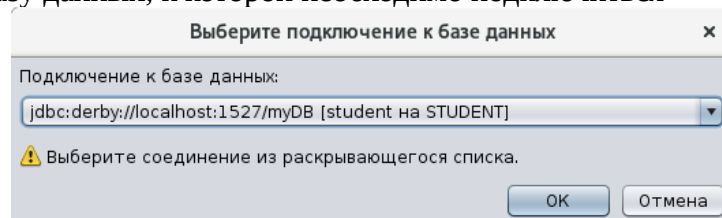
--вставка тестовых данных
insert into employee values (1, 'Ivan', 'Ivanov', 'Manager', '11-22-33');
insert into employee values (2, 'Nikolay', 'Ivanov', 'Programmer', '33-44-55');
insert into employee values (3, 'Sergey', 'Petrov', 'System administrator', '12-34-56');
insert into employee values (4, 'Alexey', 'Petrov', 'Manager', '56-78-90');
insert into employee values (5, 'Vitaliy', 'Kuznetsov', 'Technician', '55-66-77');

-- выбрать все из таблицы для проверки
select * from employee;
```

- 6) Сохраните файл.
- 7) Щелкните правой кнопкой мыши на файл employee.sql и выберите Выполнить файл



- 8) Выберите базу данных, к которой необходимо подключиться

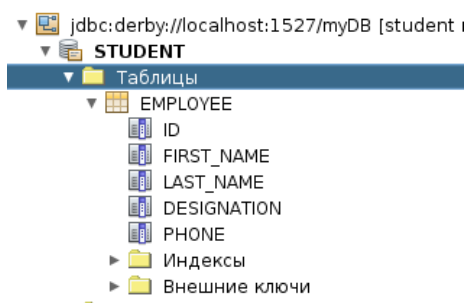


- 9) В случае успешного выполнения скрипта выводятся соответствующие сообщения и создаются таблицы базы данных, в которые заносятся данные

select * from employee x

Макс. число строк: 100 Отобранные строки: 5 Совпадающие строки:

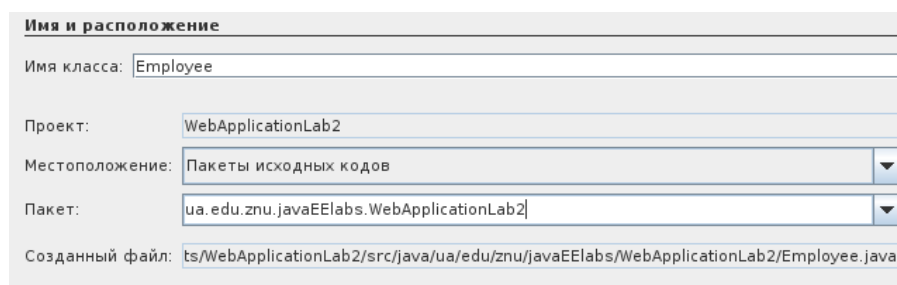
#	ID	FIRST_NAME	LAST_NAME	DESIGNATION	PHONE
1	1	Ivan	Ivanov	Manager	11-22-33
2	2	Nikolay	Ivanov	Programmer	33-44-55
3	3	Sergey	Petrov	System administrator	12-34-56
4	4	Alexey	Petrov	Manager	56-78-90
5	5	Vitaliy	Kuznetsov	Technician	55-66-77



Реализация сервлета

Перед разработкой сервлета, создадим обычный Java-класс Employee, который будет использоваться для представления и передачи данных о сотруднике.

- 1) Создайте новый Java-класс с именем Employee, нажав правой кнопкой мыши на проект WebApplicationLab2 и выбрав пункт меню Новый/Класс Java



- 2) В классе `Employee` создадим пять полей, которые соответствуют столбцам таблицы `employee`, добавим конструкторы и набор `get/set` методов. Полный код класса `Employee` приведен ниже:

```
package ua.edu.znu.javaEElabs.WebApplicationLab2;

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 создания нового сервлета щелкните правой кнопкой мыши на проект, выберите пункт меню Новый/Сервлет. Укажите пакет (Java package), в котором будет размещен сервлет. Укажите имя класса сервлета - EmployeeServlet и нажмите Готово.

Имя и расположение

Имя класса: EmployeeServlet

Проект: WebApplicationLab2

Местоположение: Пакеты исходных кодов

Пакет: ua.edu.znu.javaEElabs.WebApplicationLab2

Созданный файл: applicationLab2/src/java/ua/edu/znu/javaEElabs/WebApplicationLab2/EmployeeServlet.java

- 4) В созданный шаблон сервлета вносятся необходимые исправления, добавления и программный код (код сервлета приведен ниже). Основная бизнес логика сосредоточена в методе doGet(). Скопируйте код и сохраните сервлет.

```
package ua.edu.znu.javaEElabs.WebApplicationLab2;

import java.io.IOException;
import java.io.PrintWriter;
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 java.sql.Connection;
import java.sql.ResultSet;
import java.sql.DriverManager;
import java.util.ArrayList;

@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");
        // 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 Derby
        Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

        // Получение соединения с БД
        Connection con = DriverManager.getConnection(
            "jdbc:derby://localhost:1527/myDB;create=true;user=student;password=student");

        // 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);
}
}

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

@Override
public String getServletInfo() {
    return "Short description";
}
}

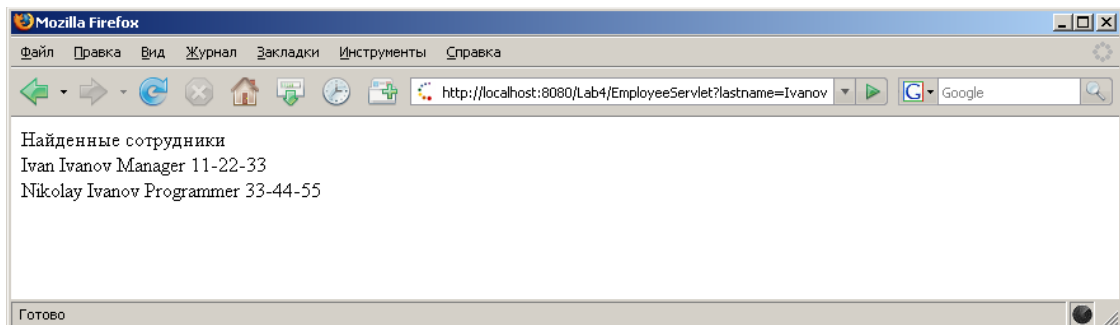
```

Тестирование приложения

Запустите браузер и перейдите по ссылке

<http://localhost:8080/Lab4/EmployeeServlet?lastname=Ivanov>

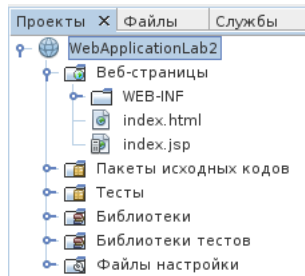
Обратите внимание, что в запросе выполняется обращение к сервлету и передается параметр `lastname` со значением `Ivanov`. Просмотрите результаты выполнения запроса.



Разработка JSP-страницы

Для улучшения функциональности приложения используются JSP-страницы. Добавим в проект одну страницу index.jsp, которая будет использоваться для ввода фамилии и отображения результатов поиска.

- 1) Для создания новой JSP-страницы, нажмите правой кнопкой мыши на проект, далее — Новый/JSP. В открывшемся диалоговом окне укажите имя файла index и нажмите Готово.



В проект был добавлен файл index.jsp. Для того чтобы этот файл использовался автоматически из проекта необходимо удалить файл index.html.

- 2) Созданный JSP-файл уже содержит базовый набор тэгов заголовка и тела веб-страницы, для решения поставленной задачи необходимо внести ряд изменений:

- изменить 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, будут являться 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.

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="ua.edu.znu.javaEElabs.WebApplicationLab2.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);

```

Тестирование приложения

Запустите браузер и перейдите по адресу <http://localhost:8080/WebApplicationLab2/>.
Введите фамилию сотрудника и нажмите «Поиск».

Фамилия сотрудника				
<input type="text" value="Petrov"/>				<input type="button" value="поиск"/>
Id	Имя	Фамилия	Должность	Телефон
3	Sergey	Petrov	System administrator	12-34-56
4	Alexey	Petrov	Manager	56-78-90

Выполните поиск, оставив поле «Фамилия сотрудника» пустым.

Задание

1. Расширить функциональные возможности функцией добавления новых сотрудников. Создайте новую страницу, содержащую форму для ввода информации о новом сотруднике и добавьте на нее ссылку на главной странице. Необходимо разработать сервлет для обработки параметров нового сотрудника и создания записи в БД. После создания сотрудника главная страница должна автоматически обновляться.
2. Расширить функциональные возможности приложения функцией изменения параметров сотрудников. Необходимо создать новую страницу, содержащую форму для изменения информации о сотруднике. Страница должна открываться при кликанье мышки на записи о сотруднике в таблице. Разработать сервлет для обработки параметров сотрудника и обновления записи в БД. После изменения атрибутов сотрудника главная страница должна автоматически обновляться.