МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

КАФЕДРА компьютерных технологий и программной инженерии

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Кочин Д.А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| «Разработка ресурса REST/JSON сервиса» |
| по курсу: Технологии разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4932 |  | 15.09.21 |  | А.И.Белов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

1. **Задание:**

Подключите к проекту Maven фреймворк Spring (spring-boot), определите перечень Rest-сервисов, выполняющих те же действия, что и в лабораторной работе 1. Внимательно отнеситесь к вопросу какой HTTP метод использует тот или иной сервис и какие коды HTTP он может возвращать. Реализуйте эти сервисы

Вариант: 2. Поликлиника (запись на прием к врачу)

1. **Описание разрабатываемого продукта:**

Веб-приложение для организации управления поликлиникой. Функционал: добавление, удаление, изменение информации о врачах и пациентах, организация записей.

Формы:

Главное окно: Изображение выглядит как текст

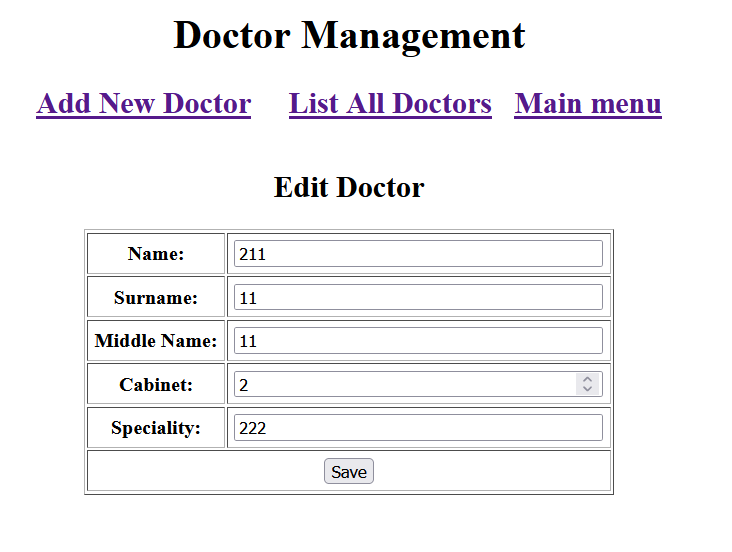
Автоматически созданное описание

Список врачей:

Изображение выглядит как стол

Автоматически созданное описание

Форма добавления/редактирования врача:

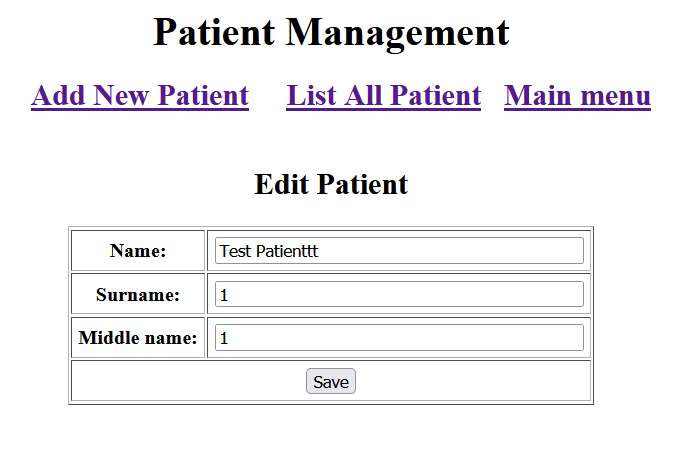


Список пациентов:

Изображение выглядит как стол

Автоматически созданное описание

SФорма добавления/редактирования пациента:

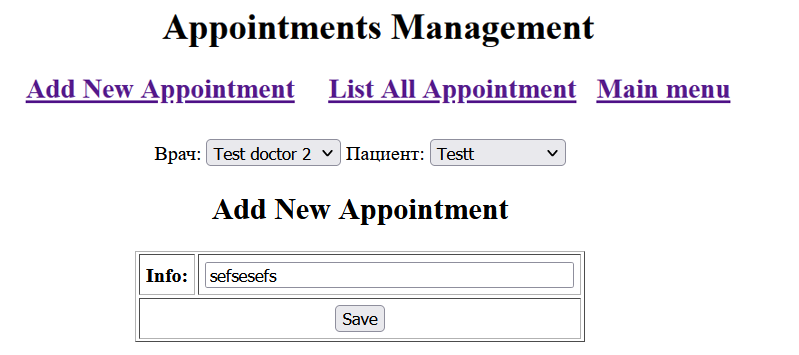


Список записей:

Изображение выглядит как стол

Автоматически созданное описание

Форма добавления записи:



1. **Листинг:**

// @filename \src\main\java\api\lab\Models\Doctor.java

package api.lab.Models;

public class Doctor {

public Integer id;

public String name;

public String surname;

public String midname;

public String speciality;

public Integer cabinet;

public Integer getId(){

return id;

}

public String getName(){

return name;

}

public String getSpeciality() {

return speciality;

}

public Integer getCabinet() {

return cabinet;

}

public void setId(Integer id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getMidname() {

return midname;

}

public void setMidname(String midname) {

this.midname = midname;

}

public void setSpeciality(String speciality) {

this.speciality = speciality;

}

public void setCabinet(Integer cabinet) {

this.cabinet = cabinet;

}

}

// @filename \src\main\java\api\lab\Models\Doctor\_Patient.java

package api.lab.Models;

import java.util.Date;

public class Doctor\_Patient {

public Integer id;

public Doctor doc;

public Patient pat;

public String info;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public Doctor getDoc() {

return doc;

}

public Patient getPat() {

return pat;

}

public String getInfo() {

return info;

}

public void setDoc(Doctor doc) {

this.doc = doc;

}

public void setPat(Patient pat) {

this.pat = pat;

}

public void setInfo(String info) {

this.info = info;

}

}

// @filename \src\main\java\api\lab\Models\Patient.java

package api.lab.Models;

import java.util.Date;

public class Patient {

public int id;

public String name;

public String surname;

public String midname;

public Integer getId(){

return id;

}

public String getName(){

return name;

}

public void setId(int id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getMidname() {

return midname;

}

public void setMidname(String midname) {

this.midname = midname;

}

}

// @filename \src\main\java\api\lab1\AppointmentDAO.java

package api.lab1;

import api.lab.Models.Doctor;

import api.lab.Models.Doctor\_Patient;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

public class AppointmentDAO {

public List<Doctor\_Patient> All() throws SQLException {

List<Doctor\_Patient> result = new ArrayList<>();

String command = "SELECT \* FROM [Doctor\_Patient]";

Statement statement = SQLCon.getSqlConnection().createStatement();

ResultSet resultSet = statement.executeQuery(command);

while(resultSet.next()){

Doctor\_Patient newAppointment = new Doctor\_Patient();

newAppointment.id= resultSet.getInt("Id");

newAppointment.pat= new PatientDAO().Get(resultSet.getInt("Id\_Patient"));

newAppointment.doc= new DoctorDAO().Get(resultSet.getInt("Id\_Doctor"));

newAppointment.info = resultSet.getString("Info");

result.add(newAppointment);

}

return result;

};

public boolean Add(Doctor\_Patient doctor\_patient) throws SQLException {

String command = "INSERT INTO Doctor\_Patient (Id\_Doctor, Id\_Patient, Info) VALUES (?,?,?)";

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setInt(1, doctor\_patient.doc.id);

statement.setInt(2, doctor\_patient.pat.id);

statement.setString(3, doctor\_patient.info);

boolean result = statement.executeUpdate()>0;

return result;

};

public boolean Delete(Doctor\_Patient doctor\_patient) throws SQLException {

String command = "DELETE FROM Doctor\_Patient WHERE Id=?";

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setInt(1,doctor\_patient.id);

boolean result = statement.executeUpdate()>0;

return result;

};

}

// @filename \src\main\java\api\lab1\DoctorDAO.java

package api.lab1;

import api.lab.Models.Doctor;

import api.lab.Models.Patient;

import javax.print.Doc;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class DoctorDAO {

public Doctor Get(int id) throws SQLException {

Doctor result = new Doctor();

String command = "SELECT \* FROM Doctor WHERE Id="+id;

Statement statement = SQLCon.getSqlConnection().createStatement();

ResultSet resultSet = statement.executeQuery(command);

if(resultSet.next()){

Doctor newDoctor = getDoctorFromResultSet(resultSet);

result = newDoctor;

}

return result;

};

public List<Doctor> All() throws SQLException {

List<Doctor> result = new ArrayList<>();

String command = "SELECT \* FROM [Doctor]";

Statement statement = SQLCon.getSqlConnection().createStatement();

ResultSet resultSet = statement.executeQuery(command);

while(resultSet.next()){

Doctor newDoctor = getDoctorFromResultSet(resultSet);

result.add(newDoctor);

}

return result;

};

public boolean Add(Doctor doctor) throws SQLException {

String command = "INSERT INTO Doctor (Name, Cabinet, Speciality, Surname, Midname) VALUES (?,?,?,?,?)";

PreparedStatement statement = getStatWithDoctor(doctor,command);

boolean result = statement.executeUpdate()>0;

return result;

};

public boolean Edit(Doctor doctor) throws SQLException {

String command = "UPDATE Doctor SET Name=?, Cabinet=?, Speciality=?, Surname=?, Midname=? WHERE Id=?";

PreparedStatement statement = getStatWithDoctor(doctor, command);

statement.setInt(6, doctor.id);

boolean result = statement.executeUpdate()>0;

return result;

};

public boolean Delete(Doctor doctor) throws SQLException {

String command = "DELETE FROM Doctor WHERE Id=?";

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setInt(1, doctor.id);

boolean result = statement.executeUpdate()>0;

return result;

};

public PreparedStatement getStatWithDoctor(Doctor doctor, String command) throws SQLException {

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setString(1, doctor.name);

statement.setInt(2, doctor.cabinet);

statement.setString(3,doctor.speciality);

statement.setString(4, doctor.surname);

statement.setString(5, doctor.midname);

return statement;

};

public Doctor getDoctorFromResultSet(ResultSet resultSet) throws SQLException {

Doctor doctor = new Doctor();

doctor.id=resultSet.getInt("Id");

doctor.name=resultSet.getString("Name");

doctor.cabinet=resultSet.getInt("Cabinet");

doctor.speciality=resultSet.getString("Speciality");

doctor.surname=resultSet.getString("Surname");

doctor.midname=resultSet.getString("Midname");

return doctor;

};

}

// @filename \src\main\java\api\lab1\HelloServlet.java

package api.lab1;

import java.io.\*;

import java.sql.ResultSet;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(name = "helloServlet", value = "/hello-servlet")

public class HelloServlet extends HttpServlet {

private String message;

public void init() {

message = "Hello World!";

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {

response.setContentType("text/html");

SQLCon con = new SQLCon();

// ResultSet data = con.getData();

// Hello

PrintWriter out = response.getWriter();

out.println("<html><body>");

out.println("<h1>" + message + "</h1>");

out.println("</body></html>");

}

public void destroy() {

}

}

// @filename \src\main\java\api\lab1\PatientDAO.java

package api.lab1;

import api.lab.Models.Patient;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class PatientDAO {

public Patient Get(int id) throws SQLException {

Patient result = new Patient();

String command = "SELECT \* FROM Patient WHERE Id="+id;

Statement statement = SQLCon.getSqlConnection().createStatement();

ResultSet resultSet = statement.executeQuery(command);

if(resultSet.next()){

Patient newPatient = getPatientFromResultSet(resultSet);

result = newPatient;

}

return result;

};

public List<Patient> All() throws SQLException {

List<Patient> result = new ArrayList<>();

String command = "SELECT \* FROM [Patient]";

Statement statement = SQLCon.getSqlConnection().createStatement();

ResultSet resultSet = statement.executeQuery(command);

while(resultSet.next()){

Patient newPatient = getPatientFromResultSet(resultSet);

result.add(newPatient);

}

return result;

};

public boolean Add(Patient patient) throws SQLException {

String command = "INSERT INTO Patient (Name, Surname, Midname) VALUES (?,?,?)";

PreparedStatement statement = getStatFromPatient(patient, command);

boolean result = statement.executeUpdate()>0;

return result;

};

public boolean Edit(Patient patient) throws SQLException {

String command = "UPDATE Patient SET Name=?, Surname=?, Midname=? WHERE Id=?";

PreparedStatement statement = getStatFromPatient(patient, command);

statement.setInt(4, patient.id);

boolean result = statement.executeUpdate()>0;

return result;

};

public boolean Delete(Patient patient) throws SQLException {

String command = "DELETE FROM Patient WHERE Id=?";

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setInt(1,patient.id);

boolean result = statement.executeUpdate()>0;

return result;

};

public Patient getPatientFromResultSet(ResultSet resultSet) throws SQLException {

Patient patient = new Patient();

patient.id=resultSet.getInt("Id");

patient.name=resultSet.getString("Name");

patient.midname=resultSet.getString("Midname");

patient.surname=resultSet.getString("Surname");

return patient;

}

public PreparedStatement getStatFromPatient(Patient patient, String command) throws SQLException {

PreparedStatement statement = SQLCon.getSqlConnection().prepareStatement(command);

statement.setString(1, patient.name);

statement.setString(2, patient.surname);

statement.setString(3, patient.midname);

return statement;

}

}

// @filename \src\main\java\api\lab1\SQLCon.java

package api.lab1;

import java.sql.\*;

public class SQLCon {

static String connectionString = "jdbc:sqlserver://127.0.0.1\\SQLEXPRESS;databaseName=PolyclinicDB;username=sa;password=test";

public SQLCon() {

String connectionUrl = "jdbc:sqlserver://127.0.0.1\\SQLEXPRESS;databaseName=PolyclinicDB;username=sa;password=test";

this.connectionString=connectionUrl;

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

public static Connection getSqlConnection() throws SQLException {

String connectionString = "jdbc:sqlserver://127.0.0.1\\SQLEXPRESS;databaseName=PolyclinicDB;username=sa;password=test";

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

Connection connection = DriverManager.getConnection(connectionString);

return connection;

}

}

// @filename \src\main\java\api\lab1\AppointmentController.java

package com.example.lab2;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.sql.SQLException;

import java.util.List;

@Controller

public class AppointmentController {

AppointmentDAO appointmentDAO = new AppointmentDAO();

@GetMapping("appointments")

public String all(Model model) throws SQLException{

List<Doctor\_Patient> list = appointmentDAO.All();

model.addAttribute("listApp", list);

return "AppointmentIndex";

}

@GetMapping("appointments/get")

public String get(Model model) throws SQLException{

Doctor\_Patient doctor\_patient = new Doctor\_Patient();

//model.addAttribute("app", doctor\_patient);

List<Doctor> listdoc = new DoctorDAO().All();

model.addAttribute("listdocs", listdoc);

List<Patient> listpat = new PatientDAO().All();

model.addAttribute("listpats", listpat);

return "AppointmentForm";

}

@PostMapping(value = "appointments/post")

public String post(@ModelAttribute Doctor\_Patient doctor\_patient, Model model) throws SQLException {

doctor\_patient.doc=new DoctorDAO().Get(doctor\_patient.iddoc);

doctor\_patient.pat=new PatientDAO().Get(doctor\_patient.idpat);

appointmentDAO.Add(doctor\_patient);

return "redirect:/appointments";

}

@GetMapping(value="appointments/delete{id}")

public String delete(@PathVariable Integer id, Model model) throws SQLException {

Doctor\_Patient doctor\_patient = new Doctor\_Patient();

doctor\_patient.id=id;

appointmentDAO.Delete(doctor\_patient);

return "redirect:/appointments";

}

}

// @filename \src\main\java\api\lab1\DoctorController.java

package com.example.lab2;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.sql.SQLException;

import java.util.List;

@Controller

public class DoctorController {

DoctorDAO doctorDAO = new DoctorDAO();

@GetMapping("doctors")

public String all(Model model) throws SQLException{

List<Doctor> list = doctorDAO.All();

model.addAttribute("listDoctor", list);

return "DoctorIndex";

}

@GetMapping(value = "doctors/get{id}")

public String get(@PathVariable Integer id, Model model) throws SQLException {

Doctor doc = doctorDAO.Get(id);

model.addAttribute("doctor", doc);

return "DoctorForm";

}

@PostMapping(value = "doctors/post")

public String post(@ModelAttribute Doctor doctor, Model model) throws SQLException {

doctorDAO.Add(doctor);

return "redirect:/doctors";

}

@PostMapping(value="doctors/edit{id}")

public String edit(@PathVariable Integer id, Doctor doctor, Model model) throws SQLException {

doctorDAO.Edit(doctor);

return "redirect:/doctors";

}

// @filename \src\main\java\api\lab1\PatientController.java

package com.example.lab2;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.sql.SQLException;

import java.util.List;

@Controller

public class PatientController {

PatientDAO patientDAO = new PatientDAO();

@GetMapping("patients")

public String all(Model model) throws SQLException{

List<Patient> list = patientDAO.All();

model.addAttribute("listPatient", list);

return "PatientIndex";

}

@GetMapping(value = "patients/get{id}")

public String get(@PathVariable Integer id, Model model) throws SQLException {

Patient patient = patientDAO.Get(id);

model.addAttribute("patient", patient);

return "PatientForm";

}

@PostMapping(value = "patients/post")

public String post(@ModelAttribute Patient patient, Model model) throws SQLException {

patientDAO.Add(patient);

return "redirect:/patients";

}

@PostMapping(value="patients/edit{id}")

public String edit(@PathVariable Integer id, Patient patient, Model model) throws SQLException {

patientDAO.Edit(patient);

return "redirect:/patients";

}

@GetMapping(value="patients/delete{id}")

public String delete(@PathVariable Integer id, Model model) throws SQLException {

patientDAO.Delete(patientDAO.Get(id));

return "redirect:/patients";

}

}

// @filename \src\main\java\api\lab1\TestController.java

package com.example.lab2;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class TestController {

@GetMapping("/")

public String greeting(@RequestParam(name="name", required=false, defaultValue="World") String name, Model model) {

model.addAttribute("name", name);

return "greeting";

}

}

// @filename \src\resources\templates\greeting.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Getting Started: Serving Web Content</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

</head>

<body>

<center>

<h1>Поликлиника имени плохого кода"</h1>

<h2>

</h2>

<table border="1" cellpadding="5">

<caption>

<h2>

Main Menu

</h2>

</caption>

<tr>

<td>

<a href="/appointments">Записи к врачу</a>

</td>

<td>

<a href="/doctors">Врачи</a>

</td>

<td>

<a href="/patients">Пациенты</a>

</td>

</tr>

</table>

</center>

</body>

</html>

// @filename \src\resources\templates\AppointmentIndex.html

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Appointments Management</h1>

<h2>

<a th:href="@{/appointments/get}">Add New Appointment</a>

&nbsp;&nbsp;&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form>

<table border="1" cellpadding="5">

<caption><h2>List of Appointment</h2></caption>

<tr>

<th>Name Doctor</th>

<th>Name Patient</th>

<th>Info</th>

</tr>

<tbody>

<tr th:each="app : ${listApp}">

<td th:text="${app.doc.name}" /></td>

<td th:text="${app.pat.name}" /></td>

<td th:text="${app.info}" /></td>

<td>&nbsp;&nbsp;&nbsp;&nbsp;

<button th:formaction="@{/appointments/delete{id}(id=${app.id})}" method="delete">Delete</button>

</td>

</tr>

</tbody>

</table>

</form>

</div>

</body>

</html>

// @filename \src\resources\templates\AppointmentForm.html

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Appointment Management</h1>

<h2>&nbsp;&nbsp;&nbsp;

<a th:href="@{/appointments}">List All Appointments</a>

&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form action="#" th:action="@{post}" method="post">

<th>Врач:</th>

<select size=1 name="iddoc" id="iddoc" required="required">

<option th:each="category: ${listdocs}" th:value="${category.id}" th:text="${category.name}"</option>

</select>

<th>Пациент:</th>

<select size=1 name="idpat" id="idpat" required="required">

<option th:each="category:${listpats}" th:value="${category.id}" th:text="${category.name}"</option>

</select>

<table border="1" cellpadding="5">

<caption>

<h2>

Add New Appointment

</h2>

</caption>

<tr>

<th>Info: </th>

<td>

<input type="text" name="info" size="45"

/>

</td>

</tr>

<tr>

<td colspan="2" align="center">

<input type="submit" value="Save"/>

</td>

</tr>

</table>

</form>

</div>

</body>

</html>

<script>

document.getElementById("iddoc").value = null;

document.getElementById("idpat").value = null;

function changeDoc(data){

document.getElementById("iddoc").value = data;

}

function changePat(data){

document.getElementById("idpat").value = data;

}

</script>

// @filename \src\resources\templates\DoctorIndex.html

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Doctors Management</h1>

<h2>

<a th:href="@{/doctors/get-1}">Add New Doctor</a>

&nbsp;&nbsp;&nbsp;

<a th:href="@{/doctors}"></a>

&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form>

<table border="1" cellpadding="5">

<caption><h2>List of Doctors</h2></caption>

<tr>

<th>Name</th>

<th>Surname</th>

<th>Middle Name</th>

<th>Cabinet</th>

<th>Speciality</th>

</tr>

<tr th:each="doctor : ${listDoctor}">

<td th:text="${doctor.name}"></td>

<td th:text="${doctor.surname}"></td>

<td th:text="${doctor.midname}"></td>

<td th:text="${doctor.cabinet}"></td>

<td th:text="${doctor.speciality}"></td>

<td>

<button th:formaction="@{/doctors/get{id}(id=${doctor.id})}">Edit</button>

&nbsp;&nbsp;&nbsp;&nbsp;

<button th:formaction="@{/doctors/delete{id}(id=${doctor.id})}" th:method="delete">Delete</button>

</td>

</tr>

</table>

</form>

</div>

</body>

</html>

// @filename \src\resources\templates\DoctorForm.html

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Doctor Management</h1>

<h2>&nbsp;&nbsp;&nbsp;

<a th:href="@{/doctors}">List All Doctors</a>

&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form action ="#" th:object="${doctor}" method="post">

<table border="1" cellpadding="5">

<caption>

<h2 th:if="${doctor.id != null}">Edit Doctor</h2>

<h2 th:if="${doctor.id == null}">Add Doctor</h2>

</caption>

<input type="hidden" name="id" th:value="${doctor.id}"/>

<tr>

<th>Name: </th>

<td>

<input type="text" name="name" size="45" th:value="${doctor.name}" />

</td>

</tr>

<tr>

<th>Surname: </th>

<td>

<input type="text" name="surname" size="45" th:value="${doctor.surname}" />

</td>

</tr>

<tr>

<th>Middle Name: </th>

<td>

<input type="text" name="midname" size="45" th:value="${doctor.midname}" />

</td>

</tr>

<tr>

<th>Cabinet: </th>

<td>

<input type="number" name="cabinet" size="45" th:value="${doctor.cabinet}" />

</td>

</tr>

<tr>

<th>Speciality: </th>

<td>

<input type="text" name="speciality" size="45" th:value="${doctor.speciality}" />

</td>

</tr>

<tr>

<td colspan="2" align="center">

<button th:if="${doctor.id==null}" type="submit" value="Add" th:formaction="@{post}" >Добавить</button>

<button th:if="${doctor.id!=null}" type="submit" value="Save" th:formaction="@{edit{id}(id=${doctor.id})}" >Сохранить</button>

</td>

</tr>

</table>

</form>

</div>

</body>

</html>

// @filename \src\resources\templates\PatientIndex.htm

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Patients Management</h1>

<h2>

<a th:href="@{/patients/get-1}">Add New Patient</a>

&nbsp;&nbsp;&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form>

<table border="1" cellpadding="5">

<thead>

<caption><h2>List of Patient</h2></caption>

<tr>

<th>Name</th>

<th>Surname</th>

<th>Middle Name</th>

</tr>

</thead>

<tbody>

<tr th:each="patient : ${listPatient}">

<td th:text="${patient.name}"></td>

<td th:text="${patient.surname}"></td>

<td th:text="${patient.midname}"></td>

<td>

<button th:formaction="@{/patients/get{id}(id=${patient.id})}">Edit</button>

&nbsp;&nbsp;&nbsp;&nbsp;

<button th:formaction="@{/patients/delete{id}(id=${patient.id})}" th:method="delete">Delete</button>

</td>

</tr>

</tbody>

</table>

</form>

</div>

</body>

</html>

// @filename \src\resources\templates\PatientForm.html

<html>

<head>

<title>Pharmacy Application</title>

</head>

<body>

<center>

<h1>Patient Management</h1>

<h2>

<a th:href="@{/patients}">List All Patients</a>

&nbsp;

<a th:href="@{/}">Main menu</a>

</h2>

</center>

<div align="center">

<form action="#" th:object="${patient}" method="post">

<table border="1" cellpadding="5">

<caption>

<h2 th:if="${patient.id != null}">Edit Patient</h2>

<h2 th:if="${patient.id == null}">Add Patient</h2>

</caption>

<tr>

<th>Name: </th>

<td>

<input type="text" name="name" size="45" th:value="${patient.name}" />

</td>

</tr>

<tr>

<th>Surname: </th>

<td>

<input type="text" name="surname" size="45" th:value="${patient.surname}" />

</td>

</tr>

<tr>

<th>Middle name: </th>

<td>

<input type="text" name="midname" size="45" th:value="${patient.midname}" />

</td>

</tr>

<tr>

<td colspan="2" align="center">

<button th:if="${patient.id==null}" type="submit" value="Add" th:formaction="@{post}">Добавить</button>

<button th:if="${patient.id!=null}" type="submit" value="Save" th:formaction="@{edit{id}(id=${patient.id})}" >Сохранить</button>

</td>

</tr>

</table>

</form>

</div>

</body>

</html>