**МИНОБРНАУКИ РОССИИ**

**САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра вычислительной техники**

**ОТЧЕТ**

**по лабораторным работам №7**

**по дисциплине «Программирование интернет-приложений на языке Java»**

**Тема: «Разработка Web-приложения с использованием GWT»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 7308 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Фомченкова Н.А. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Павловский М.Г. |

Санкт–Петербург

2021

**Цель работы**

Знакомство с процессом создания GWT-приложения в среде IntelliJ IDEA.

**Задание**

1. Создайте GWT-проект.

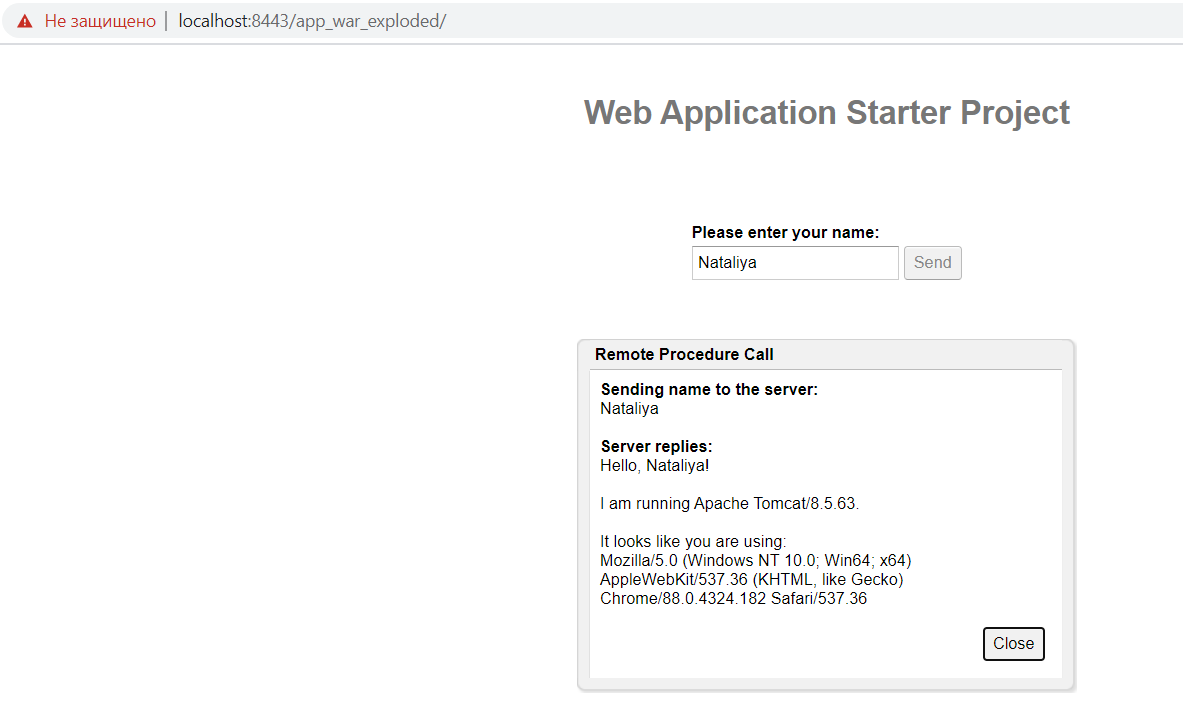
2. Используя GWT-технологию разработайте Web-приложение, функциональность которого соответствует приложению, созданному в лабораторной работе № 2.

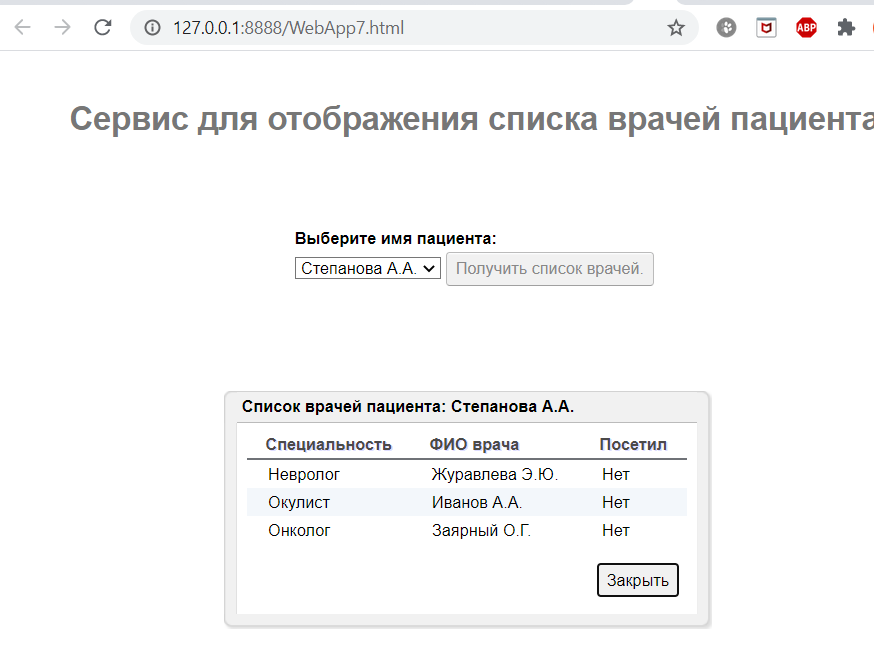
3. Скомпилируйте и запустите приложение.

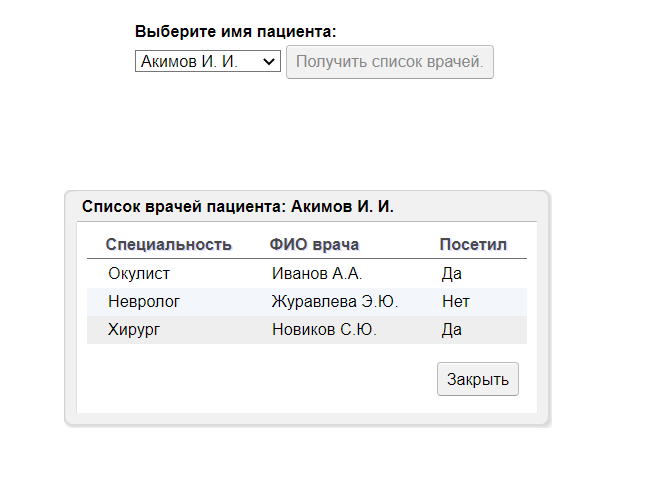
4. Протестируйте приложение и убедитесь в правильности его работы.

5. Сгенерируйте документацию с помощью Javadoc для клиентской и серверной частей приложения и просмотрите ее в браузере.

**Ход работы**







Вывод: В ходе выполнения лабораторной работы были рассмотрены способы создания GWT-приложений в среде Eclipse. Было разработано и протестировано приложение, реализованное во 2 л/р.

Листинг:

package etu.etu.web.client;  
  
import com.google.gwt.core.client.EntryPoint;  
import com.google.gwt.core.client.GWT;  
import com.google.gwt.event.dom.client.ClickEvent;  
import com.google.gwt.event.dom.client.ClickHandler;  
import com.google.gwt.event.dom.client.KeyCodes;  
import com.google.gwt.event.dom.client.KeyUpEvent;  
import com.google.gwt.event.dom.client.KeyUpHandler;  
import com.google.gwt.user.client.rpc.AsyncCallback;  
import com.google.gwt.user.client.ui.Button;  
import com.google.gwt.user.client.ui.DialogBox;  
import com.google.gwt.user.client.ui.HTML;  
import com.google.gwt.user.client.ui.Label;  
import com.google.gwt.user.client.ui.RootPanel;  
import com.google.gwt.user.client.ui.VerticalPanel;  
import etu.etu.web.shared.MedicineReader;  
import com.google.gwt.user.cellview.client.CellTable;  
import com.google.gwt.user.cellview.client.HasKeyboardSelectionPolicy.KeyboardSelectionPolicy;  
import com.google.gwt.user.cellview.client.TextColumn;  
import com.google.gwt.user.client.ui.ListBox;  
import com.google.gwt.user.client.ui.MultiWordSuggestOracle;  
import java.util.List;  
  
public class WebApp7 implements EntryPoint {  
 private static final String *SRV\_ERR* = "Ошибка сервера! ";  
 private static final String *SRV\_ERR\_GET\_READER\_LIST* = "Невозможно получить список пециентов.";  
 private static final String *SRV\_ERR\_GET\_MEDICINE\_LIST* = "Невозможно получить список врачей пациента.";  
 private static final String *GET\_MEDICINE\_LIST\_BTN* = "Получить список врачей.";  
 private static final String *CLOSE\_BTN* = "Закрыть";  
 private static final String *MEDICINE\_WND\_TITLE* = "Список врачей пациента: ";  
 */\*\* RPC-сервис \*/* private final GreetingServiceAsync greetingService = GWT  
 .*create*(GreetingService.class);  
 public void onModuleLoad() {  
 final ListBox readerListBox = new ListBox(false);  
 final MultiWordSuggestOracle oracle = new MultiWordSuggestOracle();  
 final Label errorLabel = new Label();  
 final Button sendButton = new Button(*GET\_MEDICINE\_LIST\_BTN*);  
 sendButton.addStyleName("sendButton");  
 RootPanel.*get*("readerListBoxContainer").add(readerListBox);  
 RootPanel.*get*("errorLabelContainer").add(errorLabel);  
 RootPanel.*get*("sendButtonContainer").add(sendButton);  
 readerListBox.setFocus(true);  
 greetingService.getReaderList(new AsyncCallback<List<String>>() {  
 public void onFailure(Throwable caught) {  
 errorLabel.setText(*SRV\_ERR*+*SRV\_ERR\_GET\_READER\_LIST*);  
 }  
  
 public void onSuccess(List<String> result) {  
 oracle.clear();  
 oracle.addAll(result);  
 for(String r : result){readerListBox.addItem(r);  
 }  
 }  
 });  
 final DialogBox dialogBox = new DialogBox();  
 dialogBox.setText(*MEDICINE\_WND\_TITLE*);  
 dialogBox.setAnimationEnabled(true);  
 final Button closeButton = new Button(*CLOSE\_BTN*);  
 closeButton.getElement().setId("closeButton");  
 final HTML serverResponseLabel = new HTML();  
 VerticalPanel dialogVPanel = new VerticalPanel();  
 dialogVPanel.addStyleName("dialogVPanel");  
  
 final CellTable<MedicineReader> table = createCellTable();  
 dialogVPanel.add(table);  
  
 dialogVPanel.setHorizontalAlignment(VerticalPanel.*ALIGN\_RIGHT*);  
 dialogVPanel.add(closeButton);  
 dialogBox.setWidget(dialogVPanel);  
 closeButton.addClickHandler(new ClickHandler() {  
 public void onClick(ClickEvent event) {  
 dialogBox.hide();  
 sendButton.setEnabled(true);  
 sendButton.setFocus(true);  
 }  
 });  
 class RPCClickHandler implements ClickHandler, KeyUpHandler {  
 public void onClick(ClickEvent event) {  
 sendReaderFIOToServer();  
 }  
 public void onKeyUp(KeyUpEvent event) {  
 if (event.getNativeKeyCode() == KeyCodes.*KEY\_ENTER*) {  
 sendReaderFIOToServer();  
 }  
 }  
 private void sendReaderFIOToServer() {  
 errorLabel.setText("");  
 final String readerFIO = readerListBox.getValue(readerListBox.getSelectedIndex());  
 sendButton.setEnabled(false);  
 greetingService.getMedicineReaderList(readerFIO,  
  
 new AsyncCallback<List<MedicineReader>>() {  
 public void onFailure(Throwable caught) {  
 dialogBox.setText(*SRV\_ERR*);  
 serverResponseLabel  
 .addStyleName("serverResponseLabelError");  
 serverResponseLabel.setHTML(*SRV\_ERR*+*SRV\_ERR\_GET\_MEDICINE\_LIST*);  
 dialogBox.center();  
 closeButton.setFocus(true);  
 }  
 public void onSuccess(List<MedicineReader> result) {  
 dialogBox.setText(*MEDICINE\_WND\_TITLE* + readerFIO);  
 table.setRowCount(result.size(), true);  
 table.setRowData(0, result);  
 dialogBox.center();  
 closeButton.setFocus(true);  
 }  
 });  
 }  
 }  
 RPCClickHandler handler = new RPCClickHandler();  
 sendButton.addClickHandler(handler);  
 }  
 private CellTable<MedicineReader> createCellTable() {  
 final CellTable<MedicineReader> table = new CellTable<MedicineReader>();  
 table.setKeyboardSelectionPolicy(KeyboardSelectionPolicy.*ENABLED*);  
 TextColumn<MedicineReader> authorColumn = new TextColumn<MedicineReader>() {  
 public String getValue(MedicineReader object) {  
 return object.getSpecialty();  
 }  
 };  
 table.addColumn(authorColumn, "Специальность");  
 TextColumn<MedicineReader> titleColumn = new TextColumn<MedicineReader>() {  
 public String getValue(MedicineReader object) {  
 return object.getDoctorName();  
 }  
 };  
 table.addColumn(titleColumn, "ФИО врача");  
 TextColumn<MedicineReader> isReadColumn = new TextColumn<MedicineReader>() {  
 public String getValue(MedicineReader object) {  
 return object.isVisited() ? "Да":"Нет";  
 }  
 };  
 table.addColumn(isReadColumn, "Посетил");  
  
 return table;  
 }  
}

MedicineReader.java

package etu.etu.web.shared;  
  
import java.io.Serializable;  
public class MedicineReader implements Serializable {  
 private static final long *serialVersionUID* = 1L;  
 private String specialty;  
 private String doctorName;  
 private boolean isVisited;  
  
 public MedicineReader() {  
 }  
  
 public MedicineReader(String \_specialty, String \_doctorName, boolean \_isVisited) {  
 this.isVisited = \_isVisited;  
 this.doctorName = \_doctorName;  
 this.specialty = \_specialty;  
 }  
  
 public String getSpecialty() {  
 return specialty;  
 }  
  
 public String getDoctorName() {  
 return doctorName;  
 }  
  
 public boolean isVisited() {  
 return isVisited;  
 }  
  
 public void setAuthor(String author) {  
 this.specialty = specialty;  
 }  
}

GreetingServiceImpl.java

package etu.etu.web.server;  
  
import etu.etu.web.client.GreetingService;  
import etu.etu.web.shared.MedicineReader;  
import com.google.gwt.user.server.rpc.RemoteServiceServlet;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
  
@SuppressWarnings("serial")  
public class GreetingServiceImpl extends RemoteServiceServlet implements GreetingService {  
 private Map<String, List<MedicineReader>> db = null;  
 public List<String> getReaderList() {  
 if( db == null ){  
 initDB();  
 }  
 String[] tmp = new String[db.keySet().size()];  
 db.keySet().toArray(tmp);  
 return Arrays.*asList*(tmp);  
 }  
 public List<MedicineReader> getMedicineReaderList(String readerFIO) {  
 if( db == null ){  
 initDB();  
 }  
 return db.get(readerFIO);  
 }  
 private void initDB(){  
 db = new HashMap<String, List<MedicineReader>>();  
 List<MedicineReader> entries1 = new ArrayList<MedicineReader>();  
 entries1.add(new MedicineReader("Окулист", "Иванов А.А.", true));  
 entries1.add(new MedicineReader("Невролог", "Журавлева Э.Ю.",false));  
 entries1.add(new MedicineReader("Хирург", "Новиков С.Ю.", true));  
 db.put("Акимов И. И.", entries1);  
 List<MedicineReader> entries2 = new ArrayList<MedicineReader>();  
 entries2.add(new MedicineReader("Окулист", "Иванов А.А.",false));  
 entries2.add(new MedicineReader("Терапевт", "Львова О.В.", true));  
 db.put("Лобузов Н.И.", entries2);  
 List<MedicineReader> entries3 = new ArrayList<MedicineReader>();  
 entries3.add(new MedicineReader("Невролог", "Журавлева Э.Ю.", false));  
 entries3.add(new MedicineReader("Хирург", "Новиков С.Ю.", true));  
 entries3.add(new MedicineReader("Терапевт", "Львова О.В.", true));  
 db.put("Кац А.М.", entries3);  
 List<MedicineReader> entries4 = new ArrayList<MedicineReader>();  
 entries4.add(new MedicineReader("Невролог", "Журавлева Э.Ю.", false));  
 entries4.add(new MedicineReader("Окулист", "Иванов А.А.", false));  
 entries4.add(new MedicineReader("Онколог", "Заярный О.Г.", false));  
 db.put("Степанова А.А.", entries4);  
 }  
}

GreetingService.java

package etu.etu.web.client;  
  
import com.google.gwt.user.client.rpc.RemoteService;  
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;  
import etu.etu.web.shared.MedicineReader;  
  
  
import java.util.List;  
  
*/\*\*  
 \* The client-side stub for the RPC service.  
 \*/*@RemoteServiceRelativePath("greet")  
public interface GreetingService extends RemoteService {  
 List<String> getReaderList();  
 List<MedicineReader> getMedicineReaderList(String readerFIO);  
}

GreetingServiceAsync.java

package etu.etu.web.client;  
  
import com.google.gwt.user.client.rpc.AsyncCallback;  
import etu.etu.web.shared.MedicineReader;  
import java.util.List;  
*/\*\*  
 \* The async counterpart of <code>GreetingService</code>.  
 \*/*public interface GreetingServiceAsync {  
 void getReaderList(AsyncCallback<List<String>> callback);  
 void getMedicineReaderList(String readerFIO, AsyncCallback<List<MedicineReader>> callback);  
}

JD

