**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

отчет

**по лабораторной работе № 8**

**по дисциплине** «JAVA программирование интернет приложений»

Тема: «РАЗРАБОТКА Web-ПРИЛОЖЕНИЯ

С ИСПЛЬЗОВАНИЕМ GWT»

|  |  |  |  |
| --- | --- | --- | --- |
| Студент гр. 8305 |  | Фёдоров С.С. | |
|  |  | |
| Преподаватель |  | Павловский М. Г. | |

Санкт-Петербург

2022

**Оглавление**

[**«РАЗРАБОТКА Web-ПРИЛОЖЕНИЯ С ИСПЛЬЗОВАНИЕМ GWT»** 3](#_Toc100426841)

[**Цель работы** 3](#_Toc100426842)

[**2.** **Исходный код** 4](#_Toc100426843)

[**1.** **CarsModule.gwt.xml** 4](#_Toc100426844)

[**2.** **CarsModule.java** 5](#_Toc100426845)

[**3.** **GreetingService.java** 13](#_Toc100426846)

[**4.** **GreetingServiceAsync.java** 13](#_Toc100426847)

[**5.** **GreetingServicelmpl.java** 14](#_Toc100426848)

[**6.** **CarsReader.java** 17](#_Toc100426849)

[**Вывод** 18](#_Toc100426850)

**Лабораторная работа № 8**

# **«РАЗРАБОТКА Web-ПРИЛОЖЕНИЯ С ИСПЛЬЗОВАНИЕМ GWT»**

## **Цель работы**

Знакомство с процессом создания GWT-приложения в среде Eclipce и Maven.

1. **Скриншоты, иллюстрирующие работу приложения.**

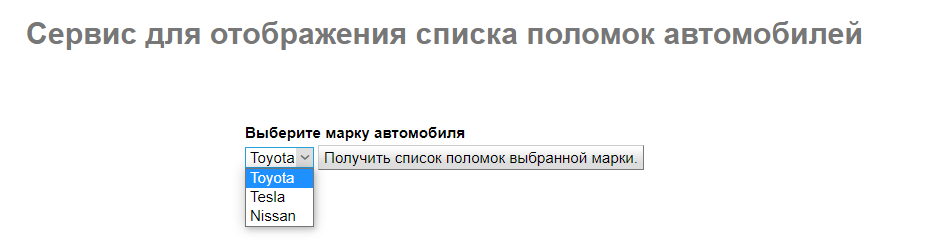


Рисунок 1 – Выбор марки автомобиля

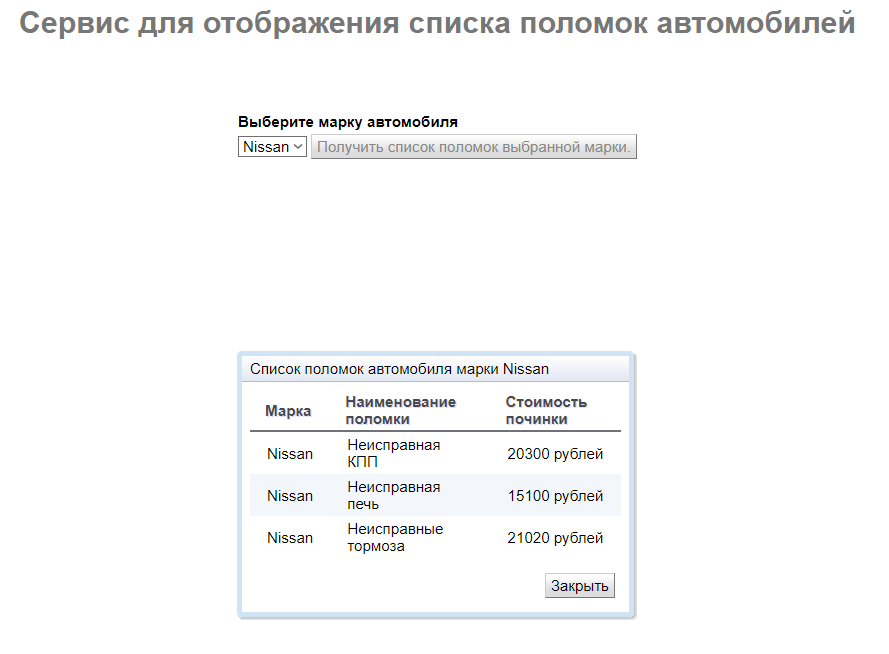


Рисунок 2 – Вывод списка поломок автомобиля «Nissan»

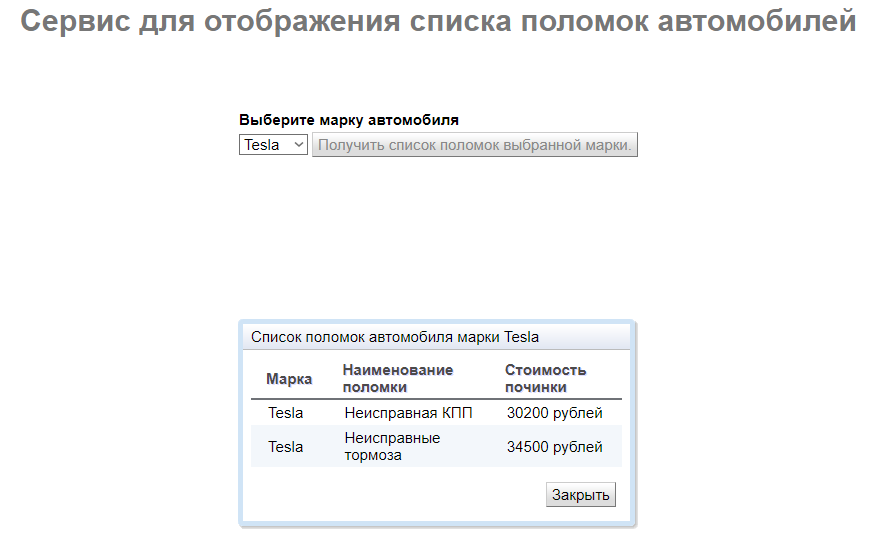


Рисунок 3 – Вывод списка поломок автомобиля «Tesla»

## **Исходный код**

### **CarsModule.gwt.xml**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**module** rename-to=*'CarsModule'*>

<!-- Inherit the core Web Toolkit stuff. -->

<**inherits** name=*'com.google.gwt.user.User'* />

<!-- Inherit the default GWT style sheet. You can change -->

<!-- the theme of your GWT application by uncommenting -->

<!-- any one of the following lines. -->

<**inherits** name=*'com.google.gwt.user.theme.standard.Standard'* />

<!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->

<!-- <inherits name='com.google.gwt.user.theme.dark.Dark'/> -->

<!-- Other module inherits -->

<!-- Specify the app entry point class. -->

<**entry-point** class=*'lab8.lab8.client.CarsModule'* />

<!-- Specify the paths for translatable code -->

<**source** path=*'client'* />

<**source** path=*'shared'* />

</**module**>

### **CarsModule.java**

package lab8.lab8.client;

import lab8.lab8.shared.FieldVerifier;

import lab8.lab8.shared.CarsReader;

import java.util.List;

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.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.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.ListBox;

import com.google.gwt.user.client.ui.MultiWordSuggestOracle;

import com.google.gwt.user.client.ui.RootPanel;

import com.google.gwt.user.client.ui.TextBox;

import com.google.gwt.user.client.ui.VerticalPanel;

/\*\*

\* Entry point classes define <code>onModuleLoad()</code>.

\*/

public class CarsModule implements EntryPoint {

/\*\*

\* The message displayed to the user when the server cannot be reached or

\* returns an error.

\*/

private static final String SERVER\_ERROR = "An error occurred while "

+ "attempting to contact the server. Please check your network "

+ "connection and try again.";

/\*\*

\* Create a remote service proxy to talk to the server-side Greeting service.

\*/

private final GreetingServiceAsync greetingService = GWT.create(GreetingService.class);

private final Messages messages = GWT.create(Messages.class);

private static final String SRV\_ERR = "Ошибка сервера! ";

private static final String SRV\_ERR\_GET\_BRANDS\_LIST = "Невозможно получить список марок автомобилей.";

private static final String SRV\_ERR\_GET\_CARS\_LIST = "Невозможно получить список поломок автомобиля марки.";

private static final String GET\_BRANDS\_LIST\_BTN = "Получить список поломок выбранной марки.";

private static final String CLOSE\_BTN = "Закрыть";

private static final String CARS\_TITLE = "Список поломок автомобиля марки ";

/\*\*

\* This is the entry point method.

\*/

public void onModuleLoad() {

final ListBox carsListBox = new ListBox(false);

final MultiWordSuggestOracle oracle = new MultiWordSuggestOracle();

final Button sendButton = new Button(GET\_BRANDS\_LIST\_BTN);

final Label errorLabel = new Label();

carsListBox.setFocus(true);

sendButton.addStyleName("sendButton");

RootPanel.get("carsListBoxContainer").add(carsListBox);

RootPanel.get("sendButtonContainer").add(sendButton);

RootPanel.get("errorLabelContainer").add(errorLabel);

greetingService.getCarsList(new AsyncCallback<List<String>>() {

public void onFailure(Throwable caught) {

errorLabel.setText (SRV\_ERR+SRV\_ERR\_GET\_BRANDS\_LIST);

}

public void onSuccess(List<String> result) {

oracle.clear();

oracle.addAll(result);

for(String r : result){carsListBox.addItem(r);}

}

});

final DialogBox dialogBox = new DialogBox();

dialogBox.setText(CARS\_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<CarsReader> table = createCellTable();

dialogVPanel.add(table);

dialogVPanel.setHorizontalAlignment(VerticalPanel.ALIGN\_RIGHT);

dialogVPanel.add(closeButton);

dialogBox.setWidget(dialogVPanel);

// Add a handler to close the DialogBox

closeButton.addClickHandler(new ClickHandler() {

public void onClick(ClickEvent event) {

dialogBox.hide();

sendButton.setEnabled(true);

sendButton.setFocus(true);

}

});

class MyHandler implements ClickHandler, KeyUpHandler {

public void onClick(ClickEvent event) {

sendNameToServer();

}

public void onKeyUp(KeyUpEvent event) {

if (event.getNativeKeyCode() == KeyCodes.KEY\_ENTER) {

sendNameToServer();

}

}

private void sendNameToServer() {

errorLabel.setText("");

final String brandName = carsListBox.getValue(carsListBox.getSelectedIndex());

sendButton.setEnabled(false);

greetingService.getCarsReaderList(brandName, new AsyncCallback<List<CarsReader>>() {

public void onFailure(Throwable caught) {

dialogBox.setText(SRV\_ERR);

serverResponseLabel.addStyleName("serverResponseLabelError");

serverResponseLabel.setHTML(SRV\_ERR+SRV\_ERR\_GET\_BRANDS\_LIST);

dialogBox.center();

closeButton.setFocus(true);

}

public void onSuccess(List<CarsReader> result) {

dialogBox.setText(CARS\_TITLE + brandName);

table.setRowCount(result.size(), true);

table.setRowData(0, result);

dialogBox.center();

closeButton.setFocus(true);

}

});

}

}

MyHandler handler = new MyHandler();

sendButton.addClickHandler(handler);

}

private CellTable<CarsReader> createCellTable() {

final CellTable<CarsReader> table = new CellTable<CarsReader>();

table.setKeyboardSelectionPolicy(KeyboardSelectionPolicy.ENABLED);

TextColumn<CarsReader> brandColumn = new TextColumn<CarsReader>() {

public String getValue(CarsReader object) {

return object.getBrand();

}

};

table.addColumn(brandColumn, "Марка");

TextColumn<CarsReader> maintananceColumn = new TextColumn<CarsReader>() {

public String getValue(CarsReader object) {

return object.getMaintanance();

}

};

table.addColumn(maintananceColumn, "Наименование поломки");

TextColumn<CarsReader> costColumn = new TextColumn<CarsReader>() {

public String getValue(CarsReader object) {

return object.getCost();

}

};

table.addColumn(costColumn, "Стоимость починки");

return table;

}

}

### **GreetingService.java**

**package lab8.lab8.client;**

**import lab8.lab8.shared.CarsReader;**

**import java.util.List;**

**import com.google.gwt.user.client.rpc.RemoteService;**

**import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;**

**/\*\***

**\* The client side stub for the RPC service.**

**\*/**

**@RemoteServiceRelativePath("greet")**

**public interface GreetingService extends RemoteService {**

**List<String> getCarsList();**

**List<CarsReader> getCarsReaderList(String brand)throws IllegalArgumentException;**

**}**

### **GreetingServiceAsync.java**

package lab8.lab8.client;

import com.google.gwt.core.client.GWT;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface GreetingServiceAsync

{

/\*\*

\* GWT-RPC service asynchronous (client-side) interface

\* @see lab8.lab8.client.GreetingService

\*/

void getCarsList( AsyncCallback<java.util.List<java.lang.String>> callback );

/\*\*

\* GWT-RPC service asynchronous (client-side) interface

\* @see lab8.lab8.client.GreetingService

\*/

void getCarsReaderList( java.lang.String brand, AsyncCallback<java.util.List<lab8.lab8.shared.CarsReader>> callback );

/\*\*

\* Utility class to get the RPC Async interface from client-side code

\*/

public static final class Util

{

private static GreetingServiceAsync instance;

public static final GreetingServiceAsync getInstance()

{

if ( instance == null )

{

instance = (GreetingServiceAsync) GWT.create( GreetingService.class );

}

return instance;

}

private Util()

{

// Utility class should not be instantiated

}

}

}

### **GreetingServicelmpl.java**

package lab8.lab8.server;

import lab8.lab8.client.GreetingService;

import lab8.lab8.shared.CarsReader;

import lab8.lab8.shared.FieldVerifier;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

/\*\*

\* The server side implementation of the RPC service.

\*/

*@SuppressWarnings*("serial")

public class GreetingServiceImpl extends RemoteServiceServlet implements

GreetingService {

private static Map<String, List<CarsReader>> *db* = null;

public List<String> getCarsList()

throws IllegalArgumentException {

if(*db* == null ){

*initDB*();

}

String[] tmp = new String[*db*.keySet().size()];

*db*.keySet().toArray(tmp);

return Arrays.*asList*(tmp);

}

public List<CarsReader> getCarsReaderList(String brand)

throws IllegalArgumentException {

if( *db* == null ){

*initDB*();

}

if (brand.equals("Nissan")||brand.equals ("Tesla") ||

brand.equals("Toyota")) return *db*.get(brand);

else throw new IllegalArgumentException ("Нет читателя "+brand);

}

public static void initDB(){

*db* = new HashMap<String, List<CarsReader>>();

List<CarsReader> entries1 = new ArrayList<CarsReader>();

entries1.add(new CarsReader("Nissan", "Неисправная КПП", "20300 рублей"));

entries1.add(new CarsReader("Nissan", "Неисправная печь", "15100 рублей"));

entries1.add(new CarsReader("Nissan", "Неисправные тормоза", "21020 рублей"));

*db*.put("Nissan", entries1);

List<CarsReader> entries2 = new ArrayList<CarsReader>();

entries2.add(new CarsReader("Tesla", "Неисправная КПП", "30200 рублей"));

entries2.add(new CarsReader("Tesla", "Неисправные тормоза", "34500 рублей"));

*db*.put("Tesla", entries2);

List<CarsReader> entries3 = new ArrayList<CarsReader>();

entries3.add(new CarsReader("Toyota", "Поломанная трансмиссия", "14100 рублей"));

entries3.add(new CarsReader("Toyota", "Большой люфт рубля", "5020 рублей"));

*db*.put("Toyota", entries3);

}

### **CarsReader.java**

package lab8.lab8.shared;

import java.io.Serializable;

public class CarsReader implements Serializable{

private static final long ***serialVersionUID*** = 1L;

private String brand;

private String maintanance;

private String cost;

public CarsReader(){

}

public CarsReader(String \_brand, String \_maintanance, String \_cost) {

this.brand = \_brand;

this.maintanance = \_maintanance;

this.cost = \_cost;

}

public String getBrand() {

return brand;

}

public String getMaintanance() {

return maintanance;

}

public String getCost() {

return cost;

}

public void setBrand(String brand) {

this.brand = brand;

}

}

## **Вывод**

В ходе выполнения данной лабораторной работы мы познакомились с процессом создания GWT-приложения.