МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

ЗВІТ

з лабораторної роботи № 6

з дисципліни

«ПРОГРАМУВАННЯ ВЕБ-ЗАСТОСУВАНЬ JAVA EE»

ВИКОНАВ

Студент групи КН-36а

Кулик В.В.

ПЕРЕВІРИВ

Доцент каф. ПІІТУ

Бабкова Н. В.

Харків 2019

Тема

««Дослідження WebSockets Java для створення розподілених додатків»»

Завдання роботи

Розробити програмну систему, яка складається з наступних модулів (назви умовні): - adminDB – web-додаток, який реалізує повний CRUD для однією з таблиць БД (можна використовувати додаток з ЛР №4).

1. adminDB – web-додаток, який реалізує повний CRUD для однією з таблиць БД (можна використовувати додаток з ЛР №4)

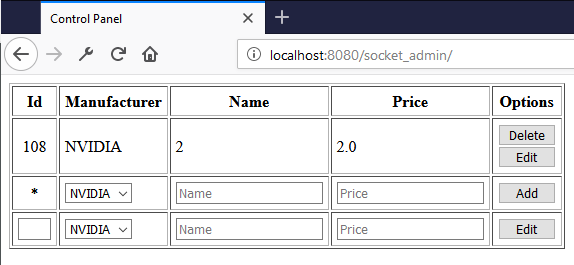
2. clientDB-web-додаток, що реалізує вибірку і фільтрацію даних з тієї ж таблиці БД. При цьому вміст сторінки даного додатки оновлюється (синхронізується) при оновленні даних (додаванні, редагуванні, оновленні) додатком adminDB. Час останнього оновлення відображається на веб-сторінці.

3. DBNetServer-консольне додаток, яке виконує наступні функції: отримує дані від adminDB при зміні вмісту БД і виводить в консолі час і тип зміни (insert, update, delete); виконує розсилку підключеним клієнтам, при отриманні clientdb оновлює вміст web-сторінки.

Хід роботи

1. Була обрана предметна область “Магазин комп’ютерних комплектуючих”. Маємо одну сутність (клас): Product. Веб-застосунок дозволяє працівникам магазину контролювати замовлення від постачальника. У замовленнях відображається назва товару, його постачальник, кількість та остаточна вартість всього замовлення. Інтерфейс дозволяє редагувати та фільтрувати дані.

Нижче представлена сторінка адміністратора:



Код JSP сторінки панелі керування адміністратора:

<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Control Panel</title>

<link rel="stylesheet" href="styles/main.css">

</head>

<body>

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

<tr>

<th>Id</th>

<th>Manufacturer</th>

<th>Name</th>

<th>Price</th>

<th>Options</th>

</tr>

<c:forEach var = "product" items="${requestScope.products}">

<tr>

<td class="id">${product.id}</td>

<td>${product.manufacturer}</td>

<td>${product.name}</td>

<td>${product.price}</td>

<td>

<form method="POST" class="deletor" action="ProductDelete">

<input hidden name="product" value="${product.id}">

<input type="submit" value="Delete">

</form>

<form method="GET" action="ProductEdit">

<input hidden name="product" value="${product.id}">

<input type="submit" value="Edit">

</form>

</td>

</tr>

</c:forEach>

<tr>

<form method="POST" id="creator" action="ProductCreate">

<td class="id">\*</td>

<td><select name="manufacturer" id="">

<option value="NVIDIA">NVIDIA</option>

<option value="ASUS">ASUS</option>

<option value="MSI">MSI</option>

</select></td>

<td><input type="text" name="name" placeholder="Name"></td>

<td><input type="text" name="price" placeholder="Price"></td>

<td><input type="submit" value="Add"></td>

</form>

</tr>

<tr>

<form method="POST" id="editor" action="ProductEdit">

<td class="id"><input type="text" readonly size="1" name="product" value="${requestScope.product.id}"></td>

<td><select name="manufacturer" id="">

<option value="NVIDIA">NVIDIA</option>

<option value="ASUS">ASUS</option>

<option value="MSI">MSI</option>

</select></td>

<td><input type="text" name="name" placeholder="Name" value="${requestScope.product.name}"></td>

<td><input type="text" name="price" placeholder="Price" value="${requestScope.product.price}"></td>

<td><input type="submit" value="Edit"></td>

</form>

</tr>

</table>

<script type="text/javascript" src="scripts/jquery-3.4.1.min.js"></script>

<script src="scripts/shop.js"></script>

<script src="scripts/admin.js"></script>

</body>

</html>

}

Код JS на сторінці адміністратора, який виконує відповідні запити:

'use strict'

var shop = (function(){

var socket = null;

var handler;

function connect(host){

if ('WebSocket' in window) {

socket = new WebSocket(host);

} else if ('MozWebSocket' in window) {

socket = new MozWebSocket(host);

} else {

console.error('WebSocket is not supported by this browser.');

return;

}

socket.onopen = function(){

console.info('WebSocket connection opened.');

}

socket.onclose = function(){

console.info('WebSocket closed.');

}

socket.onmessage = function(message){

console.log(JSON.parse(message.data));

if (handler){

handler.onmessage(message);

}

}

}

function initialize(callback){

handler = callback;

var ep = "/socket/websocket/shop";

if (window.location.protocol == "http:"){

connect('ws://' + window.location.host + ep);

} else {

connect('wss://' + window.location.host + ep);

}

}

function sendMessage(message){

if (socket) {

socket.send(JSON.stringify(message));

}

}

return {

initialize: initialize,

sendMessage: sendMessage

}

})();

Спільний для адміністратора та користувача код JS для з’єднання с сокетом:

'use strict'

var shop = (function(){

var socket = null;

var handler;

function connect(host){

if ('WebSocket' in window) {

socket = new WebSocket(host);

} else if ('MozWebSocket' in window) {

socket = new MozWebSocket(host);

} else {

console.error('WebSocket is not supported by this browser.');

return;

}

socket.onopen = function(){

console.info('WebSocket connection opened.');

}

socket.onclose = function(){

console.info('WebSocket closed.');

}

socket.onmessage = function(message){

console.log(JSON.parse(message.data));

if (handler){

handler.onmessage(message);

}

}

}

function initialize(callback){

handler = callback;

var ep = "/socket/websocket/shop";

if (window.location.protocol == "http:"){

connect('ws://' + window.location.host + ep);

} else {

connect('wss://' + window.location.host + ep);

}

}

function sendMessage(message){

if (socket) {

socket.send(JSON.stringify(message));

}

}

return {

initialize: initialize,

sendMessage: sendMessage

}

})();

Код классу з веб-сокетом на сервері:

package socket.endpoints;

import java.io.IOException;

import java.util.Date;

import java.util.List;

import java.util.concurrent.CopyOnWriteArrayList;

import javax.websocket.CloseReason;

import javax.websocket.EncodeException;

import javax.websocket.EndpointConfig;

import javax.websocket.OnClose;

import javax.websocket.OnMessage;

import javax.websocket.OnOpen;

import javax.websocket.Session;

import javax.websocket.server.ServerEndpoint;

import org.hibernate.SessionFactory;

import com.google.gson.Gson;

import net.bytebuddy.asm.Advice.ArgumentHandler.Factory;

import socket.coders.MessageDecoder;

import socket.coders.MessageEncoder;

import socket.models.Message;

import socket.models.Product;

import socket.models.Repository;

@ServerEndpoint(value = "/websocket/shop", encoders = MessageEncoder.class, decoders = MessageDecoder.class)

public class ShopEndpoint {

static private List<ShopEndpoint> clients = new CopyOnWriteArrayList<ShopEndpoint>();

static public Repository repository;

static Gson gson = new Gson();

static public List<Product> products;

Session session;

@OnOpen

public void onOpen(Session session, EndpointConfig endpointConfig) {

this.session = session;

clients.add(this);

}

@OnClose

public void onClose(Session session, CloseReason reason) {

System.out.println("Socket closed: " + reason.getReasonPhrase());

clients.remove(this);

}

@OnMessage

public void onMessage(Message msg) throws EncodeException {

switch (msg.getType()) {

case 1:

System.out.println("delete, " + new Date() + " ");

break;

case 2:

System.out.println("create, " + new Date() + " ");

break;

case 3:

System.out.println("edit, " + new Date() + " ");

break;

default:

break;

}

List<Product> products = repository.getProducts();

for (ShopEndpoint client : clients) {

try {

client.session.getBasicRemote().sendText((gson.toJson(products)));

} catch (IOException e) {

clients.remove(this);

try {

client.session.close();

} catch (IOException e1) {

e1.printStackTrace();

}

}

}

}

}

Код сторінки користувача, яка відображає в реальному часі усі зміни у таблиці Products:

<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Shop</title>

<link rel="stylesheet" href="styles/main.css">

</head>

<body>

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

<tr>

<th>Id</th>

<th>Manufacturer</th>

<th>Name</th>

<th>Price</th>

</tr>

<c:forEach var="product" items="${requestScope.products}">

<tr class="content">

<td class="id">${product.id}</td>

<td>${product.manufacturer}</td>

<td>${product.name}</td>

<td>${product.price}</td>

</tr>

</c:forEach>

</table>

<form action="Admin" method="POST">

<lable for="manufacturerPico">Filter by manufacturer:</lable>

<select id="manufacturerPico" name="manufacturer">

<option value="NVIDIA">NVIDIA</option>

<option value="ASUS">ASUS</option>

<option value="MSI">MSI</option>

</select>

<input type="submit" value="Filter">

</form>

<a href="/socket\_user">Show all</a>

<span id="time"></span>

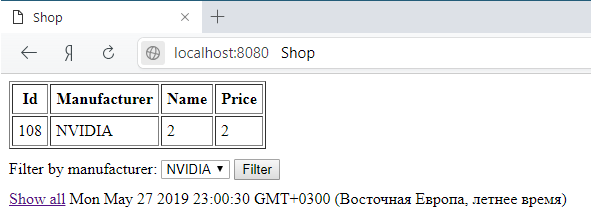
<script src="scripts/shop.js"></script>

<script src="scripts/user.js"></script>

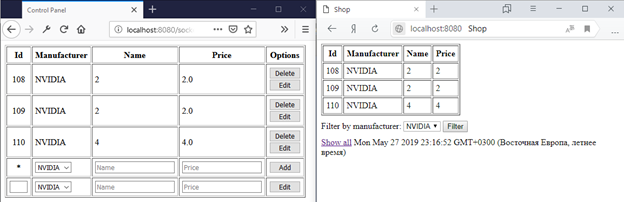
</body>

</html>

Відповідна сторінка користувача, на якій окрім таблиці і фільтра є останній час її оновлення:



Демонстрація роботи веб-сокетів адміністратор-клієнт:



**Висновок**: на даній лабораторній роботі були вдосконалені навички роботи зі стеком технологій для створення веб-додатку на Java, за допомогою сокетів та серверу Tomcat було створено декілька веб-застосунків. Програма для користувачів оновлюється у реальному часі, в залежності від змін, які вносить адміністратор за допомогою реалізації веб-сокетів на сервері.