

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Компонентні технології програмного забезпечення»**

**на тему:**

**«**Розробка та використання компонентів, керованих повідомленнями.**»**

Виконав:

Студент гр. ПЗ2011

Кулик С.В.

Прийняв:

Андрющенко В. О.

Дніпро, 2023

**Тема.** Розробка та використання компонентів, керованих повідомленнями.

**Теоретичні відомості**

Eclipse Foundation перейменувала Java EE в Jakarta EE у 2018. Це набір специфікацій та документації для мови Java, що описує архітектуру серверної платформи для середніх і великих підприємств. Ці специфікації забезпечують переносимість програм між різними платформами і масштабованість додатків. MDB - це частина JEE, яка використовує анотації для спрощення створення компонентів, керованих повідомленнями. Обмін повідомленнями - це механізм комунікації, який дозволяє програмам обмінюватися даними незалежно від мови програмування або операційної системи. JMS - це Java API, який підтримує обмін повідомленнями за моделлю «однорангові» і «публікувати/підписувати». MDB - це компонент, що викликається контейнером при надходженні повідомлення в систему.

**Текст програми**

**Файл HelloServlet.java**

package com.example.lr2;  
import java.io.\*;  
import javax.jms.\*;  
import javax.naming.Context;  
import javax.naming.InitialContext;  
import javax.naming.NamingException;  
import javax.servlet.http.\*;  
import javax.servlet.annotation.\*;  
  
@WebServlet(name = "servlet", value = "/servlet")  
public class HelloServlet extends HttpServlet {  
 private String message;  
  
 public void init() {  
 message = "Hello World!";  
 }  
  
 @Override  
 protected void doGet(  
 HttpServletRequest req,  
 HttpServletResponse res)  
 throws IOException {  
 String text = req.getParameter("text") != null ? req.getParameter("text") : message;  
 try {  
 Context ic = new InitialContext();  
 ConnectionFactory cf = (ConnectionFactory) ic.lookup("/ConnectionFactory");  
 Queue queue = (Queue) ic.lookup("queue/tutorialQueue");  
 Connection connection = cf.createConnection();  
 Session session = connection.createSession(false, Session.*AUTO\_ACKNOWLEDGE*);  
 MessageProducer publisher = session.createProducer(queue);  
 connection.start();  
 TextMessage message = session.createTextMessage(text);  
 publisher.send(message);  
 } catch (NamingException | JMSException e) {  
 res.getWriter()  
 .println("Error while trying to send \"" + text + "\", message: " + e.getMessage());  
 }  
 res.getWriter().println("Message sent: " + text);  
 }  
}

**Файл ReadMessageMDB.java**

package com.example.lr2;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.MessageDriven;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

import javax.servlet.annotation.WebListener;

@MessageDriven(activationConfig = {

@ActivationConfigProperty(

propertyName = "destination",

propertyValue = "tutorialQueue"),

@ActivationConfigProperty(

propertyName = "destinationType",

propertyValue = "javax.jms.Queue")

})

@WebListener

public class ReadMessageMDB implements MessageListener {

public void onMessage(Message message) {

TextMessage textMessage = (TextMessage) message;

try {

System.*out*.println("Message received: " + textMessage.getText());

} catch (JMSException e) {

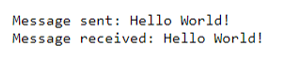
System.*out*.println("Error while trying to consume messages: " + e.getMessage());

}

}

}

**Результати роботи програми**



**Висновок**

Під час проведення лабораторної роботи ми досліджували тему розробки та використання компонентів, що працюють на основі повідомлень. Ми ознайомилися з концепціями, пов'язаними з обміном повідомленнями та різними типами комунікації, включаючи синхронний та асинхронний підхід. Також ми вивчили Java Message Services ("JMS"), який є Java API для обміну повідомленнями і забезпечує рівень "точка до точки" та "видавець/підписник". Ми також дослідили компоненти, керовані повідомленнями (MDB), які дозволяють створювати рішення зі слабким зв'язком та взаємодіяти між різними мовами програмування та операційними системами. В результаті цієї лабораторної роботи ми отримали глибше розуміння основ розробки компонентів, що працюють на основі повідомлень, і їх застосування в різних областях.

Крім того, на стороні клієнта ми розробляємо клієнтський додаток, який взаємодіє з сервером через мережу. Клієнт викликає методи, що визначені на сервері, і отримує результати виконання цих методів.