Міністерство науки та освіти України

Національний Університет «Львівська Політехніка»

Кафедра інформаційних систем та мереж

Лабораторна робота №5

з дисципліни «Екстремальне програмування»

Виконав:

Студент групи КН-311

Куренда Т.І.

Прийняв:

Щербак С.С.

Львів-2020

**Мета роботи:** основи роботи з Apache Maven і Apache Tomcat.

**Теоретичні відомості**

Apache Tomcat — контейнер сервлетів, розроблений Apache Software Foundation. Повністю написаний мовою програмування Java та реалізує специфікацію сервлетів і Java Server Pages від Sun Microsystems, що є стандартами для розробки веб-застосунків на Java.

Члени ASF і незалежні добровольці розвивають та підтримують Tomcat. Користувачі мають вільний доступ до вихідного коду Tomcat за умовами Apache License. Першою релізною версією Tomcat стала версія 3.0.x (попередні версії були випущені Sun для внутрішнього користування).

«Apache Maven» — це засіб автоматизації роботи з програмними проектами, який спочатку використовувався для Java проектів. Використовується для управління (management) та складання (build) програм. Створений 2002 року Джейсоном ван Зилом. За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML. XML-файл описує проект, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни. Сервер із додатковими модулями та додатковими бібліотеками розміщується на серверах. Раніше Maven був частиною Jakarta Project.

Для опису програмного проекту, який потрібно побудувати (build), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач — таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).

Ключовою особливістю Maven є його мережева готовність (network-ready).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних Java-проектів з відкритим кодом, від Apache та інших організацій та окремих розробників. Цей репозиторій та його реорганізований наступник, — Maven 2 репозиторій, — намагається бути де-факто механізмом для дистрибуції Java програм, але прийняття його в такій якості йде повільно.

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє як первісний засіб синхронізації виходу проектів на локальній системі.

Maven базується на плагін-архітектурі, що дозволяє зробити використання будь-якої програми контрольованим через стандартний вхід. Теоретично, це могло б дозволити будь-кому писати плагіни для інтерфейсу з інструментами для побудови (компілятори, тестери тощо) для будь-якої мови. В дійсності, підтримка і використання для мов відмінних від Java були мінімальною. Тепер існують плагіни для .NET та C/C++.

**Приклад**

Maven конфігурує проекти за допомогою конструкції ProjectObjectModel, що зберігається в файлі pom.xml. Найпростіший приклад цього файлу:



**Виконання завдання**

package ee;  
  
import BD.Pipe;  
  
import javax.servlet.RequestDispatcher;  
import javax.servlet.ServletConfig;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
public class log\_in extends HttpServlet {  
  
 @Override  
 public void init(ServletConfig config) throws ServletException {  
 super.init(config);  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 RequestDispatcher requestDispatcher;  
  
 String uri = req.getRequestURI();  
 String params = getParameter(req);  
 ArrayList<Person> person = new ArrayList<>();  
 int age = 0;  
 String name = "";  
 String Password="";  
 for (Map.Entry<String, String[]> map : req.getParameterMap().entrySet()) {  
  
 if (map.getKey().equals("Username")) {  
 name = String.*join*("and", map.getValue());  
  
 }  
 if (map.getKey().equals("Password")) {  
 Password = String.*join*("and", map.getValue());  
 }  
 }  
  
 Pipe pipe = new Pipe();  
 /\* person.add(new Person())\*/  
  
 requestDispatcher = req.getRequestDispatcher("/output.jsp");  
 requestDispatcher.forward(req, resp);  
 }  
  
 private String getParameter(HttpServletRequest req) {  
 return req.getParameterMap().entrySet().stream().map(  
 stringEntry -> {  
 String param = String.*join*(" and ", stringEntry.getValue());  
 param=stringEntry.getKey() + " param====>" + param;  
 return param;  
  
 }).collect(Collectors.*joining*("\n"));  
 }  
}

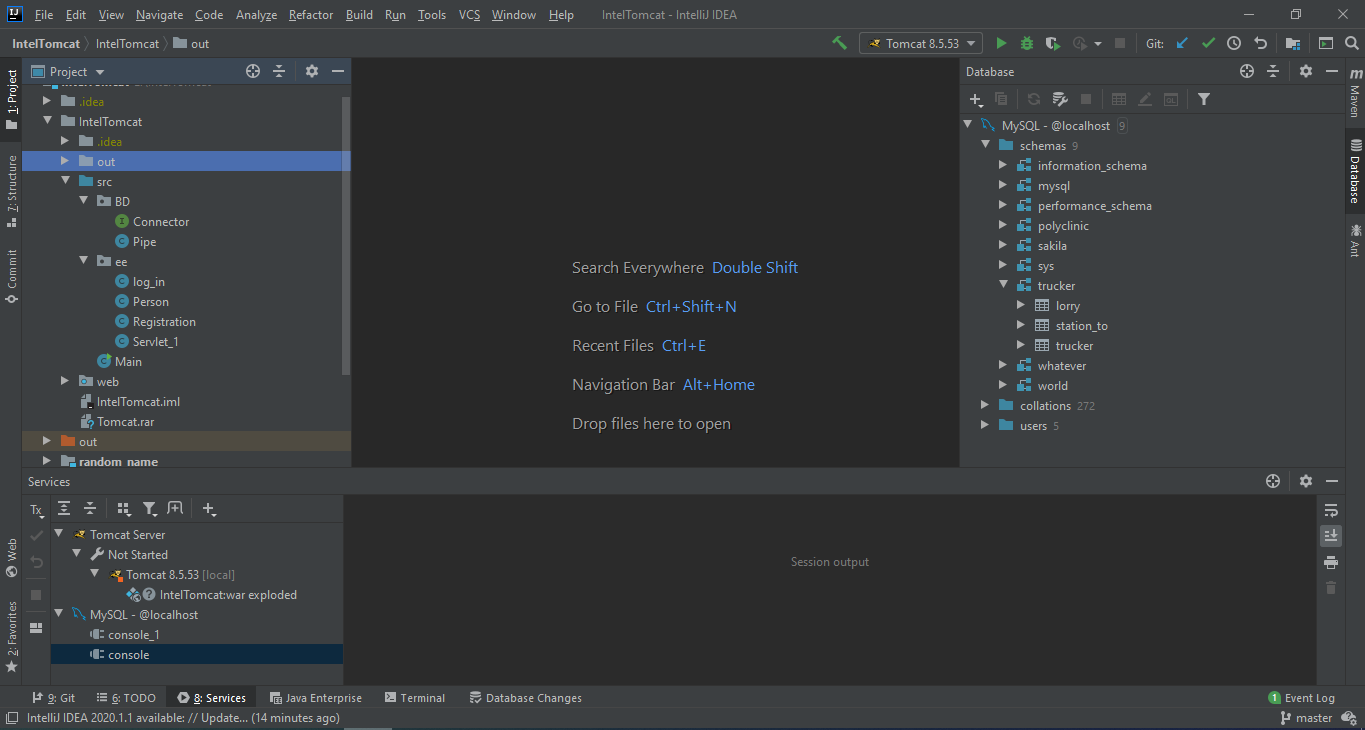
package ee;  
  
public class Person {  
 String Full\_name;  
 int age;  
 String password;  
  
// constructor for setting name and age  
 public Person(String full\_name, int age) {  
 Full\_name = full\_name;  
 this.age = age;  
 }  
  
  
 public void setFull\_name(String full\_name) {  
  
 Full\_name = full\_name;  
 }  
  
 public void setAge(int age) {  
  
 if (age >= 0) {  
  
 this.age = age;  
 } else {  
 System.*out*.println("Вік не може бути від'ємний");  
 }  
 }  
  
 public String getFull\_name() {  
 return Full\_name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 // overload of constructor  
 Person(String password,String full\_name){  
 this.password = password;  
 Full\_name = full\_name;  
 }

package ee;  
  
import BD.Pipe;  
  
 import javax.servlet.RequestDispatcher;  
 import javax.servlet.ServletConfig;  
 import javax.servlet.ServletException;  
 import javax.servlet.annotation.WebServlet;  
 import javax.servlet.http.HttpServlet;  
 import javax.servlet.http.HttpServletRequest;  
 import javax.servlet.http.HttpServletResponse;  
 import java.io.IOException;  
 import java.sql.SQLException;  
 import java.util.ArrayList;  
 import java.util.Map;  
 import java.util.stream.Collectors;  
  
@WebServlet("/registration")  
public class Registration extends HttpServlet {  
  
  
 @Override  
 public void init(ServletConfig config) throws ServletException {  
 super.init(config);  
 }  
  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 RequestDispatcher requestDispatcher = req.getRequestDispatcher("/REgistration.jsp");  
 requestDispatcher.forward(req, resp);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.*out*.println(" sdasdadsasd asdjka sdjao;s dhaposudapsudy aousdhpa9 su yhpoiasudyapoisu yaoiusda");  
  
 ArrayList<Person> person = new ArrayList<>();  
 int age = 0;  
 String name = "";  
 String Password = "";  
 for (Map.Entry<String, String[]> map : req.getParameterMap().entrySet()) {  
  
 if (map.getKey().equals("Username")) {  
 name = String.*join*("and", map.getValue());  
  
 }  
 if (map.getKey().equals("Password")) {  
 Password = String.*join*("and", map.getValue());  
  
 }  
  
 }  
 Pipe pipe = new Pipe();  
  
 try {  
 pipe.connect(name, Password);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 RequestDispatcher requestDispatcher = req.getRequestDispatcher("/cabinet.jsp");  
 requestDispatcher.forward(req, resp);  
  
 }  
  
 private String getParameter(HttpServletRequest req) {  
 return req.getParameterMap()  
 .entrySet()  
 .stream()  
 .map(stringEntry -> {  
 String param = String.*join*("and", stringEntry.getValue());  
 return stringEntry.getKey() + "=>>" + param;  
 }).collect(Collectors.*joining*("\n"));  
 }

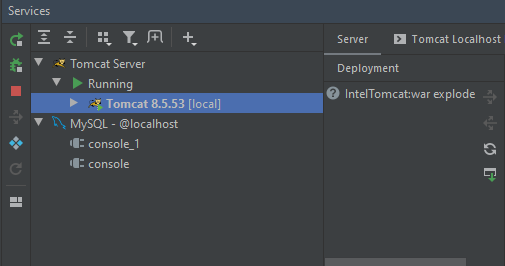
package ee;  
  
import javax.servlet.RequestDispatcher;  
import javax.servlet.ServletConfig;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Map;  
import java.util.stream.Collectors;  
import java.sql.\*;  
  
import BD.Connector;  
import BD.Pipe;  
  
public class Servlet\_1 extends HttpServlet {  
  
 @Override  
 public void init(ServletConfig config) throws ServletException {  
 super.init(config);  
 }  
  
  
 @Override  
 protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 super.service(req, resp);  
  
  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String uri = req.getRequestURI();  
 String params = getParameter(req);  
 resp.getWriter().write("url =>>" + uri + " param" + params);  
 resp.getWriter().write(" Hello ");  
 ArrayList<Person> person = new ArrayList<>();  
 int age = 0;  
 String name = "";  
 for (Map.Entry<String, String[]> map : req.getParameterMap().entrySet()) {  
  
 if (map.getKey().equals("Age")) {  
 age = Integer.*parseInt*(String.*join*("", map.getValue()));  
  
 }  
 if (map.getKey().equals("Name")) {  
 name = String.*join*("and", map.getValue());  
  
 }  
 }  
 Pipe pipe = new Pipe();  
 try {  
 pipe.connect(name,age);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 person.add(new Person(name, age));  
  
  
 req.setAttribute("people", person);  
  
 /\* person.add(new Person())\*/  
 RequestDispatcher requestDispatcher = req.getRequestDispatcher("/output.jsp");  
 requestDispatcher.forward(req, resp);  
  
  
 }  
  
 private String getParameter(HttpServletRequest req) {  
 return req.getParameterMap()  
 .entrySet()  
 .stream()  
 .map(stringEntry -> {  
 String param = String.*join*("and", stringEntry.getValue());  
 return stringEntry.getKey() + "=>>" + param;  
 }).collect(Collectors.*joining*("\n"));  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
  
 }  
  
 @Override  
 public void destroy() {  
  
 }

import BD.Pipe;  
  
import java.sql.SQLException;  
  
public class Main {  
int inter;  
 public static void main(String[] args) throws SQLException {  
  
Main obj =new Main();  
 System.*out*.println(obj.inter);  
 }  
}

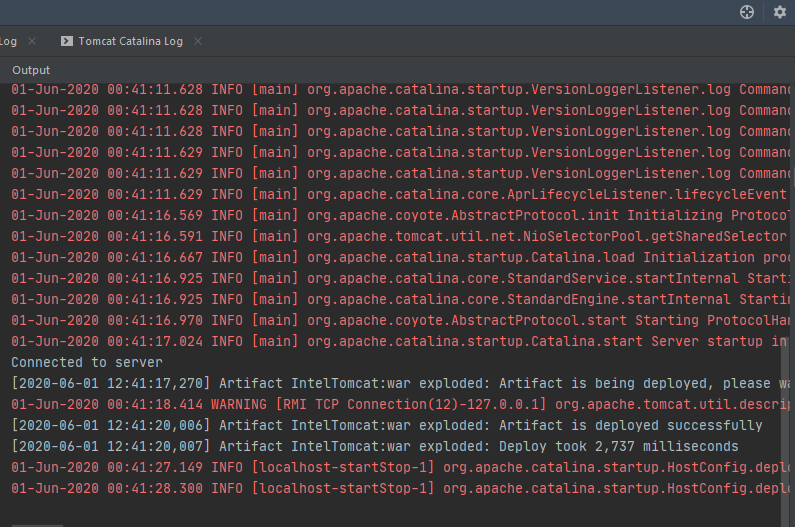
Структура проекту:



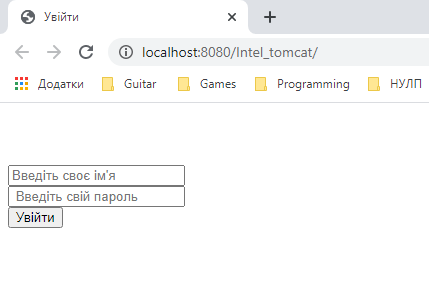
Запускаємо Tomcat Server:



Тут бачимо логи:



Результат успішної роботи сервлету та Apache Tomcat.



**Висновок:** під час виконання лабораторної роботи я навчився основам користування Apache Tomcat для написання проекту.