

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

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

**на тему:**

**«**Розробка та використання сеансових JavaBean-компонентів.**»**

Виконала:

Студент гр. ПЗ2011

Кулик С.В.

Прийняв:

Андрющенко В. О.

Дніпро, 2023

**Тема.** Розробка та використання сеансових JavaBean-компонентів.

**Теоретичні відомості**

***Entity-Компонент*** є об'єктним поданням даних, що знаходяться в довготривалому сховищі, такому, як бази даних, або даних, реалізованих за допомогою наявної програми. Вони призначені для того, щоб кілька клієнтів використовували їх одночасно, на відміну від Session-компонентів, які завжди взаємодіють тільки з одним клієнтом.

***Entity-Компонент*** співпадає із записом (або набором записів) в таблиці реляційної бази даних або одиничного об'єкту в об'єктно-орієнтованій базі даних. Наприклад, в разі використання реляційних баз даних, кожен стовпець запису в таблиці відповідає полям Entity-компонента. Кожен запис в таблиці ідентифікується за допомогою головного ключа - одного або декількох стовпців, які унікальним чином ідентифікують кожен запис. Точно також, кожен Entity-компонент містить головний ключ (primary key) для ідентифікації конкретного екземпляра компонента.

Хоча кілька клієнтів можуть взаємодіяти з одним компонентом одночасно, Контейнер EJB управляє цією взаємодією таким чином, щоб зберегти цілісність інформації в базі даних. Контейнер може управляти конкурентним доступом до одного компоненту, наприклад, за допомогою вибудовування запитів клієнтів в чергу так, щоб тільки один запит виконувався в кожен конкретний час. Контейнер може також делегувати управління конкурентним доступом засобам управління базами даних (DBMS) - тобто контейнер тільки створює екземпляри компонентів для кожного клієнта, а потім повністю покладається на DBMS.

Так як Entity-компонент зазвичай є представленням інформації в базі даних, його цикл життя збігається з циклом життя самих даних - Entity-компонент існує протягом довгого часу, зазвичай набагато довше, ніж клієнтська програма, яка його створила. Такі компоненти існують не тільки після завершення сеансу зв'язку або клієнтської програми, вони існують довше, ніж серверний процес, в контексті якого вони виконуються. Подібно даним в базі даних, Entity-компонент не руйнується в результаті збою сервера з подальшим його перезапуском.

На відміну від Session-компонентів, з якими використовується тайм-аут (і для яких контейнер видаляє екземпляр компонента при завершенні періоду тайм- ауту), до Entity-компонентів неможливо вживати поняття тайм-ауту. Незалежно від того, як довго вони залишаються неактивними, контейнер не видаляє їх зі сховища. Контейнер видаляє тільки екземпляри компонента з самого контейнери.

Єдиний спосіб видалити Entity-компонент - зробити це явно. Це може бути зроблено за допомогою виклику методу remove (), який видаляє як екземпляр Компонента, так і зіставлену з ним інформацію з бази даних, або за допомогою засобів DBMS (або інших існуючих програм, які не мають відношення до EJB).

**Текст програми**

**Файл IUserHello.java**

package com.example.lab4;  
  
import javax.ejb.Local;  
@Local  
public interface IUserHello  
{  
 public String sayHello(String name);  
}

**Файл UserHelloBean.java**

package com.example.lab4;  
  
import javax.ejb.Stateless;  
@Stateless  
public class UserHelloBean implements IUserHello  
{  
 @Override  
 public String sayHello(String name) {  
 return String.*format*("The user %s is logged in ", name);  
 }  
}

**Файл Servlet.java**

package com.example.lab4;

import javax.ejb.EJB;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "Servlet", urlPatterns = "/Servlet")

public class Servlet extends HttpServlet {

@EJB

private IUserHello hello;

private static final long *serialVersionUID* = 1L;

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {

try {

String answer = hello.sayHello(request.getParameter("login"));

request.getSession().setAttribute("answer", answer);

RequestDispatcher rd = request.getRequestDispatcher("index.jsp");

rd.forward(request, response);

} catch (Exception e) {

throw new ServletException(e.getMessage());

}

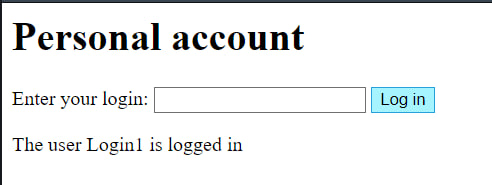
}

}

**Файл index.jsp**

<html>  
<head>  
 <title>Personal account</title>  
</head>  
<body>  
<h1>Personal account</h1>  
<form method="post" action="Servlet">  
 <label for="login">Enter your login:</label>  
 <input type="text" id="login" name="login" required>  
 <input type="submit" value="Log in">  
</form>  
<**c:if** test="${not empty answer}">  
 <p>${answer}</p>  
</**c:if**>  
</body>  
</html>

**Результати роботи програми**



Можна побачити що програма працює коректно, та виконує поставлені завдання.

**Висновок**

JavaBean-компоненти - це зручний спосіб розробки повторно використовуваних компонентів у Java-програмуванні, які можна використовувати для розробки графічниї інтерфейсів, сервлетів, моделі даних і т.д. Їх стандартні методи доступу до властивостей дозволяють налаштовувати поведінку компонента, а методи для виконання певних дій дозволяють використовувати їх як незалежні компоненти в більш складних системах. Використання JavaBean-компонентів забезпечує більшу перевикористовуваність коду, модульність, зниження складності та більшу гнучкість у веб-розробці. В загальному, вони є потужним інструментом для розробки високоякісних програм, що забезпечують ефективну взаємодію між різними компонентами.