Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники (ИИТР)**

**Лабораторная работа № 2**

**по дисциплине**

**«Распределённые программные системы»**

Тема «Сессионные компоненты»

Выполнила:

ст. гр. ИСТ-117

Шумейко Д.С.

Принял:

Макаров С.А.

Владимир, 2020

ЦЕЛЬ РАБОТЫ

Освоить создание сессионных компонентов EJB 3.1 и использование сессионных компонентов в веб-приложении с помощью CDI.

ВЫПОЛНЕНИЕ РАБОТЫ

В синглтоне считается количество сессий всего приложения (можно проверить, запустив в двух браузерах).

package ru.rps.lr1.ejb.component;  
  
import javax.ejb.LocalBean;  
import javax.ejb.Singleton;  
import javax.inject.Named;  
  
@LocalBean  
@Named("applicationBean")  
@Singleton  
public class Application {  
 private int sessionsCount;  
  
 public void addSession() {  
 sessionsCount++;  
 }  
  
 public int getSessionsCount() {  
 return sessionsCount;  
 }  
}

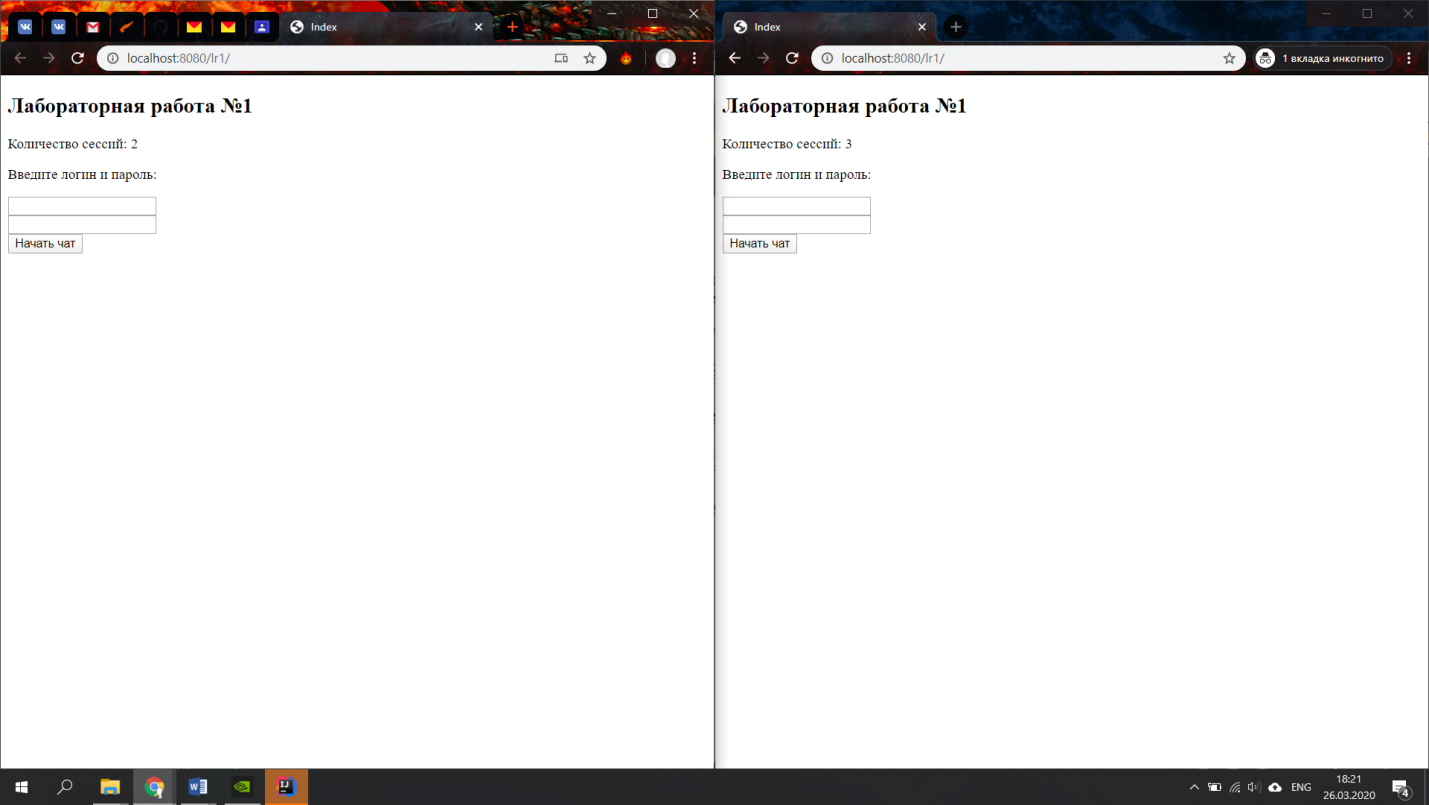


Рис.1.

Компонент без состояния LoginUtils проверяет логин и пароль – user / password.

package ru.rps.lr1.ejb.component;  
  
import javax.ejb.Stateless;  
  
@Stateless  
public class LoginUtils implements LoginUtilsLocal {  
 @Override  
 public boolean checkCredentials(String login, String password) {  
 return login.equals("user") && password.equals("password");  
 }  
}

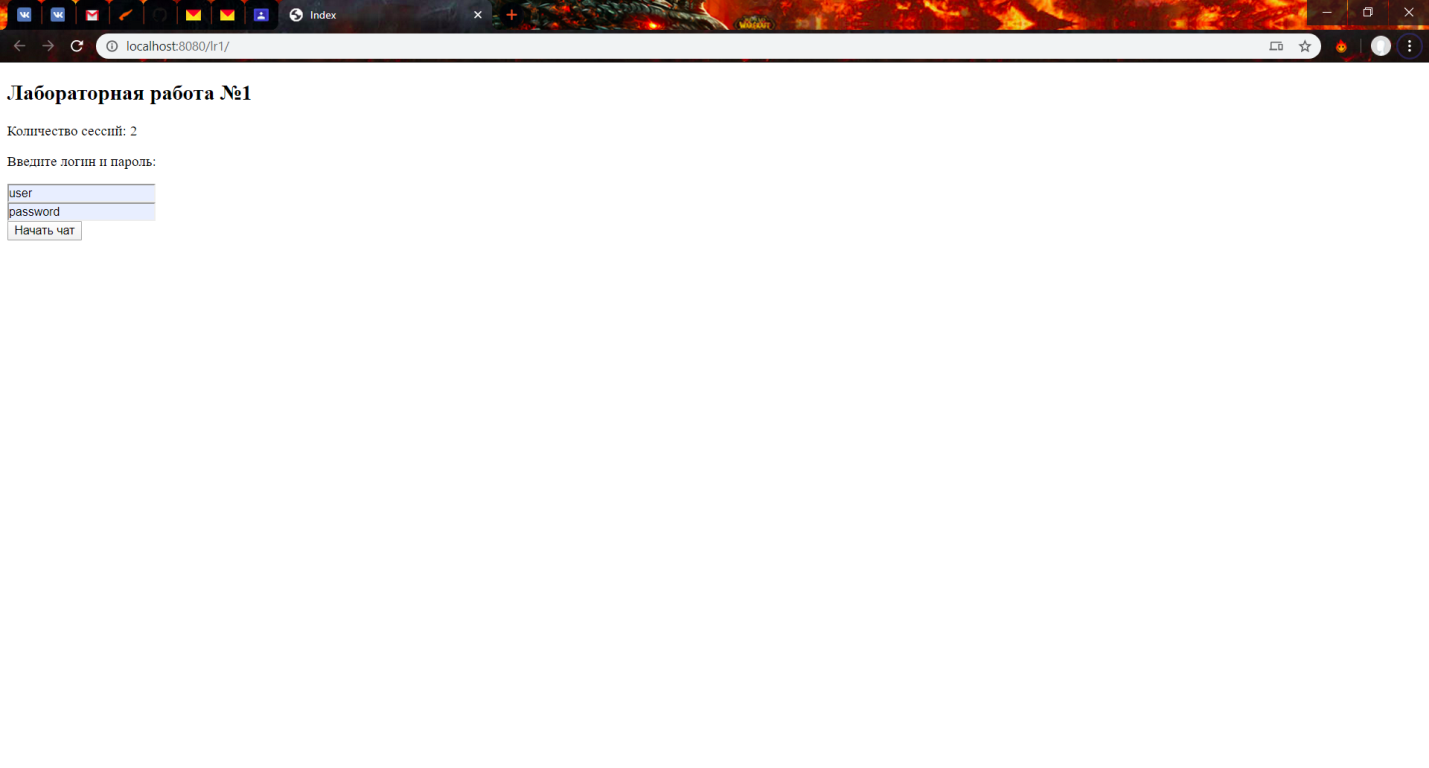


Рис.2.

Компонент без состояния ReplyUtils повторяет сообщение, введенное пользователем.

package ru.rps.lr1.ejb.component;  
  
import javax.ejb.Stateless;  
import javax.interceptor.Interceptors;  
  
import ru.rps.lr1.ejb.interceptor.EventInterceptor;  
  
@Interceptors(EventInterceptor.class)  
@Stateless  
public class ReplyUtils implements ReplyUtilsRemote {  
 @Override  
 public String reply(String message) {  
 return message;  
 }  
}

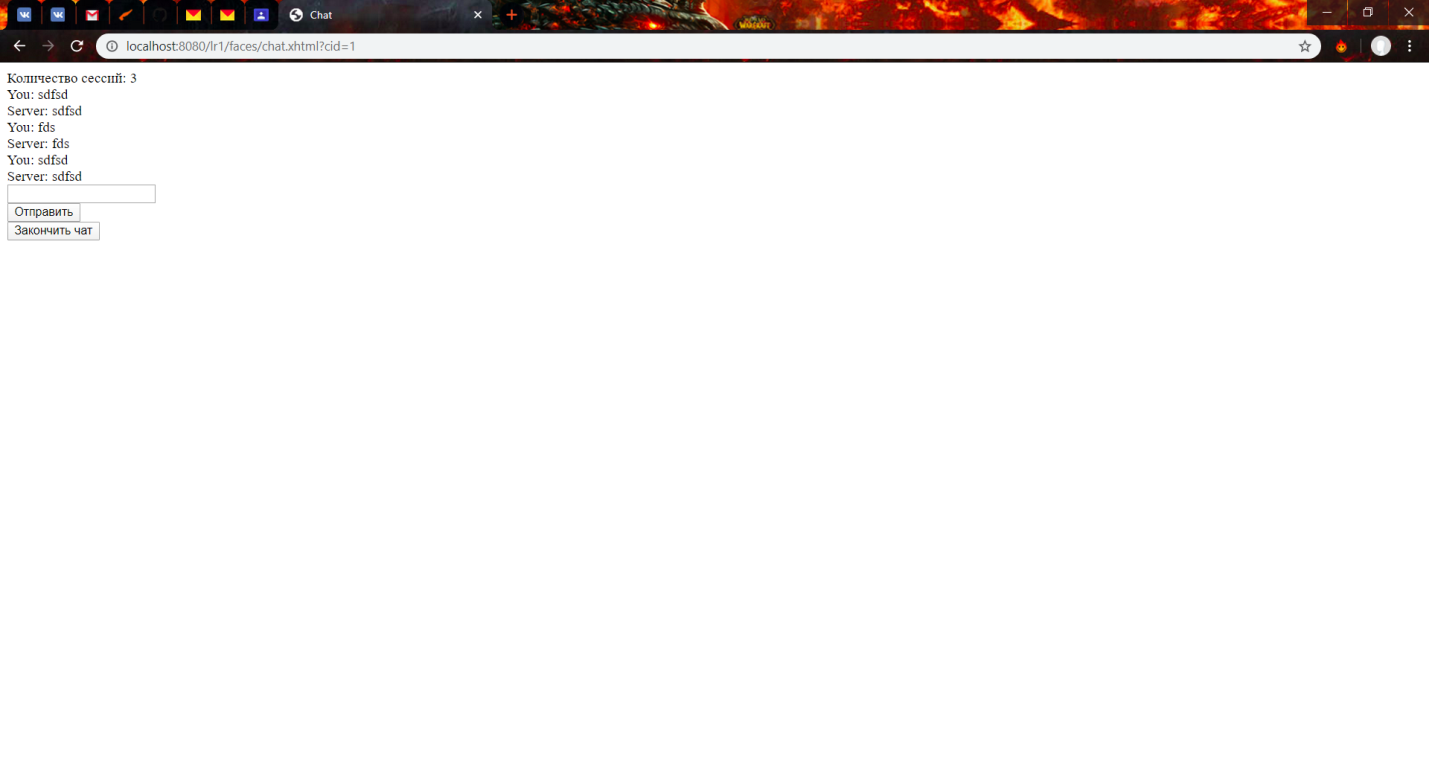


Рис.3.

Компонент с состоянием SessionState области SessionScope хранит имя пользователя.

package ru.rps.lr1.web.component;  
  
import java.io.Serializable;  
import javax.annotation.PostConstruct;  
import javax.ejb.EJB;  
import javax.ejb.Stateful;  
import javax.enterprise.context.SessionScoped;  
import javax.inject.Named;  
  
import ru.rps.lr1.ejb.component.Application;  
  
@SessionScoped  
@Stateful  
public class SessionState implements SessionStateLocal, Serializable {  
  
 @EJB  
 private Application application;  
  
 private String currentUserName;  
 private boolean loggedIn;  
  
 @PostConstruct  
 private void postConstructMethod() {  
 application.addSession();  
 }  
  
 @Override  
 public void setCurrentUser(String userName) {  
 currentUserName = userName;  
 loggedIn = true;  
 }  
  
 @Override  
 public String getCurrentUser() {  
 return currentUserName;  
 }  
  
 @Override  
 public boolean isLoggedIn() {  
 return loggedIn;  
 }  
}

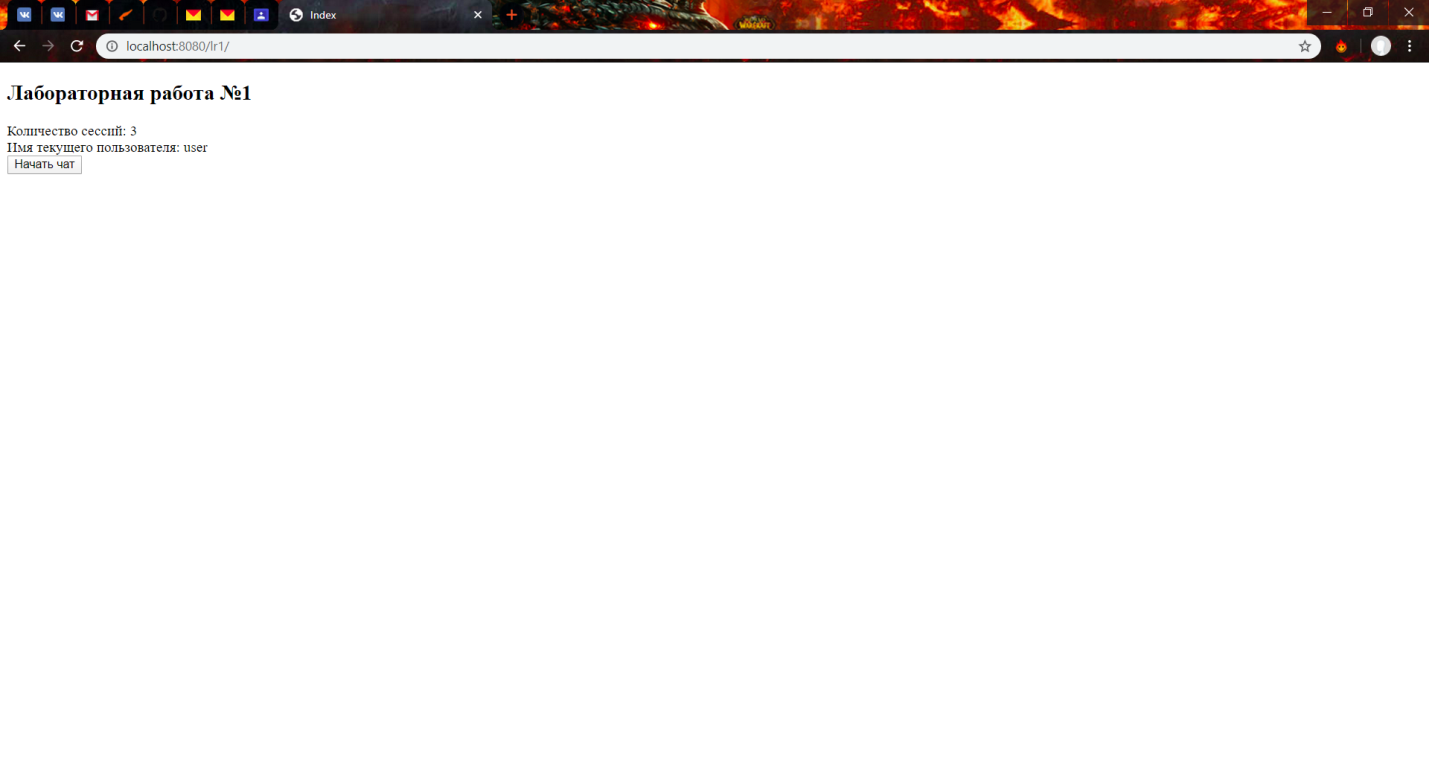


Рис.4.

Компонент с состоянием ChatState области ConversationScope хранит сообщения чата.

package ru.rps.lr1.web.component;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
import javax.ejb.Stateful;  
import javax.enterprise.context.Conversation;  
import javax.enterprise.context.ConversationScoped;  
import javax.inject.Inject;  
  
@ConversationScoped  
@Stateful  
public class ChatState implements ChatStateLocal, Serializable {  
  
 @Inject  
 private Conversation conversation;  
 private List<String> messages;  
  
 public ChatState() {  
 messages = new ArrayList<>();  
 }  
  
 @Override  
 public void start() {  
 conversation.begin();  
 }  
  
 @Override  
 public void save(String message) {  
 messages.add(message);  
 }  
  
 @Override  
 public List<String> history() {  
 return messages;  
 }  
  
 @Override  
 public void stop() {  
 conversation.end();  
 }  
}

Перехватчик (EventInterceptor) пишет сообщения в лог сервера.

package ru.rps.lr1.ejb.interceptor;  
  
import java.time.LocalDateTime;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.annotation.PostConstruct;  
import javax.interceptor.AroundInvoke;  
import javax.interceptor.Interceptor;  
import javax.interceptor.InvocationContext;  
  
@Interceptor  
public class EventInterceptor {  
 private static Logger *log* = Logger.*getLogger*("EventInterceptor");  
  
 @PostConstruct  
 public void logCreation(InvocationContext cnt) {  
 String objectName = cnt.getTarget().toString();  
 *log*.log(Level.*INFO*, String.*format*("Created %s, Time: %tT", objectName, LocalDateTime.*now*()));  
 try {  
 cnt.proceed();  
 } catch (Exception e) {  
 *log*.log(Level.*SEVERE*, String.*format*("Exception: %s", e));  
 }  
 }  
  
 @AroundInvoke  
 public Object logInvocation(InvocationContext cnt) throws Exception {  
 StringBuilder params = new StringBuilder();  
 for(Object parameter : cnt.getParameters()) {  
 params.append(parameter.toString());  
 }  
 *log*.log(Level.*INFO*, String.*format*("Method %s was invoked with parameters: %s", cnt.getMethod().getName(), params.toString()));  
 return cnt.proceed();  
 }  
}

ВЫВОДЫ

В ходе выполнения данной лабораторной работы было освоено создание сессионных компонентов EJB 3.1 и использование сессионных компонентов в веб-приложении с помощью CDI.