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)**

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 05**

**по дисциплине**

**«Распределённые программные системы»**

**Тема «Основы каркаса JSF»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Трифонов Д.А.

Владимир, 2019

ЦЕЛЬ РАБОТЫ

Изучить базовые возможности каркаса JavaServer Faces для разработки веб-приложений (организация интерфейса пользователя, обработка событий, навигация) и получить навыки реализации веб-приложений с использованием данного каркаса.

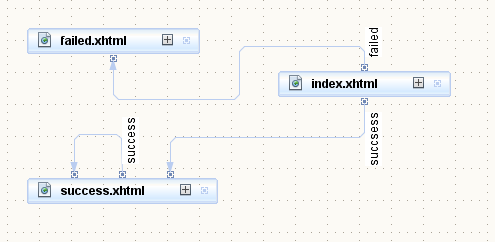
ВЫПОЛНЕНИЕ РАБОТЫ

Разработать веб-приложение, состоящее из двух страниц: одна страница должна обеспечивать ввод данных, а вторая – отображение результатов их обработки и возврат на первую страницу. Данные должны быть трех типов: строковые, числовые и дата/время. В приложении должна быть предусмотрена проверка (валидация) входных данных: строковых - на соответствие шаблону, числовых – на принадлежность диапазону.

// Конфигурационный файл. Содержит настройки (область жизни: request – создаёт объект бина на каждый запрос, session – создаётся один раз, в начале работы и используется всю сессию) бинов и навигацию.

1. Faces-config.xml

*<?***xml version='1.0' encoding='UTF-8'***?>*<**faces-config version="2.2"  
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_2.xsd"**>  
 <**managed-bean**>  
 <**description**>Обработка данных о клиенте</**description**>  
 <**managed-bean-name**>client</**managed-bean-name**>  
 <**managed-bean-class**>client.Client</**managed-bean-class**>  
 <**managed-bean-scope**>request</**managed-bean-scope**>  
 </**managed-bean**>  
 <**managed-bean**>  
 <**description**>Класс контекста сессии</**description**>  
 <**managed-bean-name**>sessionBean</**managed-bean-name**>  
 <**managed-bean-class**>sessionBean</**managed-bean-class**>  
 <**managed-bean-scope**>session</**managed-bean-scope**>  
 </**managed-bean**>  
 <**navigation-rule**>  
 <**from-view-id**>/index.xhtml</**from-view-id**>  
 <**navigation-case**>  
 <**from-outcome**>succsess</**from-outcome**>  
 <**to-view-id**>/success.xhtml</**to-view-id**>  
 </**navigation-case**>  
 <**navigation-case**>  
 <**from-outcome**>failed</**from-outcome**>  
 <**to-view-id**>/failed.xhtml</**to-view-id**>  
 </**navigation-case**>  
 </**navigation-rule**>  
 <**navigation-rule**>  
 <**from-view-id**>/success.xhtml</**from-view-id**>  
 <**navigation-case**>  
 <**from-outcome**>success</**from-outcome**>  
 <**to-view-id**>/success.xhtml</**to-view-id**>  
 </**navigation-case**>  
 </**navigation-rule**>  
</**faces-config**>



//Создаются два бина(Client, sessionBean) с областью существования request и session соответственно.

1. ManagedBean

**package** client;  
  
**import** java.util.ArrayList;  
**import** java.util.Date;  
  
**public class** Client {  
 // пароль для перехода на страницу, где производятся расчёты.  
 **private** String password;

**// дата, которую вводит пользователь**  
 **private** Date significantDay;

**// Сообщение, которое видит пользователь, после обработки даты.**  
 **private static** String numbDay = **"(￢\_￢)"**;  
 //гетеры и сетеры для созданных полей.  
 **public** Date getSignificantDay() {  
 **return** significantDay;  
 }   
 **public void** setSignificantDay(Date significantDay) {  
 **this**.significantDay = significantDay;  
 }  
 **public** String getNumbDay() {  
 **return** numbDay;  
 }   
   
 **public** String getPassword() {  
 **return** password;  
 }   
 **public void** setPassword(String password) {  
 **this**.password = password;  
 }  
 //метод, обрабатывающий пароль. Если пароль верный, показывается страница, где можно произвести расчёты, если пароль не верен, то показывается стр. ошибки.   
 **public** String checkLogin(){  
 **if** (password.equalsIgnoreCase(**"111"**)){  
 **return "success"**;  
 } **else** {  
 **return "failed"**;  
 }  
 }

**//метод, обрабатывающий дату и выводящий соответствующее сообщение (событие уже наступило, или до события х дней) При вызове методы, страница обновляется.**  
 **public** String length(){  
 Date date = **new** Date();  
 **if** (significantDay.after(date)){  
 **long** time = significantDay.getTime()- date.getTime();  
 time = time/(24\*60\*60\*1000)+1;  
 **if** (time==1) {numbDay = **"До события: "**+time+**" день"**; **return "success"**;}  
 **if** ((time>1)&&(time<5)) {numbDay = **"До события: "**+time+**" дня"**; **return "success"**;}   
 **if** (time>4) {numbDay = **"До события: "**+time+**" дней"**; **return "success"**;}   
 }  
 **if** (significantDay.before(date)){  
 numbDay = **"Событие уже прошло"**; **return "success"**;  
 }  
 numbDay = **"День Х сегодня!!!"**;  
 **return "success"**;  
 }

**//Поля для математических вычислений и соответствующие им гет/сет методы.**   
 **private int** slag1;  
 **private int** slag2;  
 **private double** result;//переменная в которую записывается результат вычислений  
   
 **public void** setSlag1(**int** slag1) {  
 **this**.slag1 = slag1;  
 }   
 **public void** setSlag2(**int** slag2) {  
 **this**.slag2 = slag2;  
 }   
 **public double** getResult() {  
 **return** result;  
 }  
 **public int** getSlag1() {  
 **return** slag1;  
 }  
 **public int** getSlag2() {  
 **return** slag2;  
 }  
 //методы для математических вычислений. При вызове метода текущая страница обновляется.  
 **public** String sum(**int** slag1, **int** slag2){  
 result = slag1+slag2;  
 **return "success"**;  
 }  
   
 **public** String dif(**int** slag1, **int** slag2){  
 result = slag1-slag2;  
 **return "success"**;  
 }  
   
 **public** String mult(**int** slag1, **int** slag2){  
 result = slag1\*slag2;  
 **return "success"**;  
 }  
   
 **public** String sub(**int** slag1, **int** slag2){  
 result = (**double**)slag1/(**double**)slag2;  
 **return "success"**;  
 }  
}

**public class** sessionBean {

**//поля в которых содержится кол-во посещений страницы** и логин пользователя(имя которое он ввёл на одной странице при «входе»)  
 **private static int** numb;  
 **private** String login;  
   
 **public** String getLogin() {  
 **return** login;  
 }  
 **public void** setLogin(String login) {  
 **this**.login = login;  
 }  
 //при обращении к странице, на которой находится данная переменная, её кол-во увеличивается.  
 **public int** getNumb(){  
 **return** numb++;  
 }   
}

1. Представления

*<?***xml version='1.0' encoding='UTF-8'** *?>***<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"*>***<**html xmlns="http://www.w3.org/1999/xhtml"  
 xmlns:h="http://xmlns.jcp.org/jsf/html"  
 xmlns:f="http://java.sun.com/jsf/core"** > //подключение библиотек тегов  
 <**h:head**>  
 <**title**>Страница авторизации</**title**>  
 </**h:head**>  
 <**h:body**>

*//создание HTML формы на странице*  
 <**h:form**>

*//Аналог таблицы.* Свойство **columns отвечает за кол-во столбцов**, далее идёт стиль таблицы: цвет, ширина и размер текста   
 <**h:panelGrid columns="1" bgcolor="#FF9B73" width="400" style="font-size**: 20**px**;**"**>  
 <**h:outputText value="Кол-во обращений к странице: #{sessionBean.numb}"** />   
 </**h:panelGrid**>  
 </**h:form**>   
 <**h:form**>  
 <**h:panelGrid columns="2" bgcolor="#FF9B73" width="400" style="font-size**: 20**px**;**"**>  
 <**h:outputText value="Введите логин"**/>  
 <**h:inputText value="#{sessionBean.login}" validatorMessage="Введите логин на английском языке   
 с большой буквы, в имени не должны присутствовать символы цифр,   
 длина текста от 3 до девяти символов"**>  
 <**f:validateLength minimum="3" maximum="9"** />  
 <**f:validateRegex pattern='[A-Z]{1}[a-z]\*'** />  
 </**h:inputText**>  
 <**h:outputText value="Введте пароль"** />  
 <**h:inputSecret value="#{client.password}" validatorMessage="длина пароля максимум три символа"**>  
 <**f:validateLength maximum="3"** />  
 </**h:inputSecret**>  
 <**h:commandButton value="GO!" style="font-size**: 15**px**;**" action="#{client.checkLogin()}"**/>  
 </**h:panelGrid**>  
 </**h:form**>  
 </**h:body**>  
</**html**>

*Комментарии по тегам:*

<h:outputText> – вывод на страницу символов/значений. В теге присутствует атрибут value которому присваивается значение, которое отразится на странице.

<h:inputText> – поле ввода, атрибут validatorMessage отвечает за валидацию(если валидация этого поля провалена, то выводится сообщение, указанное в данном атрибуте)

<f:validateLeng> - проверка введённой строки на длину. Указывается максимальная и минимальная длина.  
 <f:validateRegex> - проверка введённой строки, на соответствие шаблону(регулярному выражению.) В моём премере следующее: <f:validateRegex pattern='[A-Z]{1}[a-z]\*' /> - Это значит что имя пользователя должно начинается с большой буквы. В имени должна присутствовать одна заглавная буква и могут быть использованы только латинские знаки.

<h:inputSecret> - поле ввода для вводы пароля. Знаки скрываются символами.

<h:commandButton> - отображение кнопки, атрибут action отвечает за событие(метод), которое произойдет после нажатия на кнопку.

<f:convertDateTime> - тег, конвертирует введённую строку в дату.(Используется в следующем представлении.)

<h:link> - ссылка для перехода на указаннаю страницу. Атрибут outcome указывает на какую страницу следует перейти. (Используется в двух следующих представлениях.)

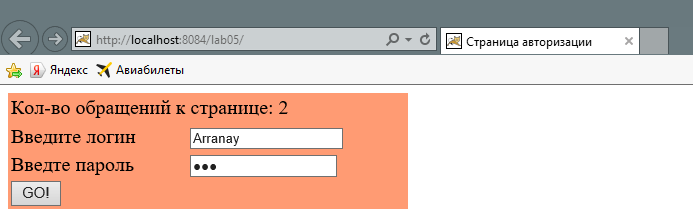


Рисунок 1. Пример первой страницы.

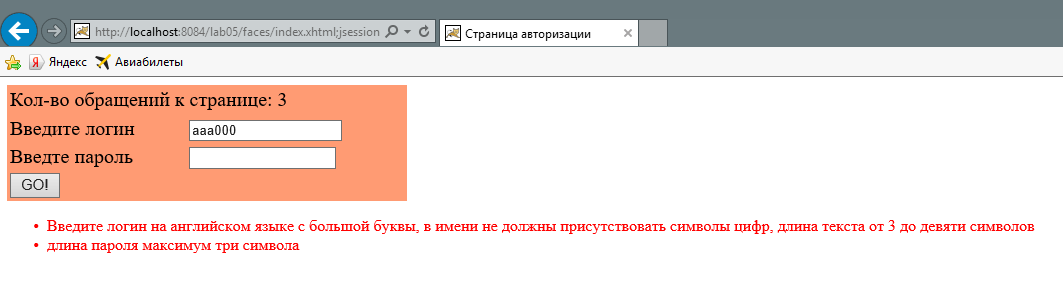


Рисунок 2. Пример валидации на первой странице.

*<?***xml version='1.0' encoding='UTF-8'** *?>***<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"*>***<**html xmlns="http://www.w3.org/1999/xhtml"  
 xmlns:h="http://xmlns.jcp.org/jsf/html"  
 xmlns:f="http://java.sun.com/jsf/core"** >  
 <**h:head**>  
 <**title**>Успешная авторизация</**title**>  
 </**h:head**>  
 <**h:body**>  
 <**h:form**>   
 <**h:panelGrid columns="1" bgcolor="#CC8AF3" width="600" style="font-size**: 20**px**;**"**>  
 <**h:outputText value="#{sessionBean.login}, добро пожаловать"** />   
 </**h:panelGrid**>  
 </**h:form**>   
   
 <**h:form**>   
 <**h:panelGrid columns="4" bgcolor="#D6A6F3" width="600" style="font-size**: 20**px**;**"**>   
 <**h:outputText value="Введите дату события:"** />  
 <**h:inputText value="#{client.significantDay}" validatorMessage="Введите дату по шаблону день.месяц.год"**>  
 <**f:validateLength minimum="10" maximum="10"** />  
 <**f:convertDateTime dateStyle="short"** />  
 </**h:inputText**>  
 <**h:commandButton value="GO!" action="#{client.length()}"**/>  
 <**h:outputText value="#{client.numbDay}"** />  
 </**h:panelGrid**>  
 </**h:form**>   
   
 <**h:form**>  
 <**h:panelGrid columns="8" bgcolor="#CC8AF3" width="600" style="font-size**: 20**px**;**"**>  
 <**h:inputText value="#{client.slag1}" validatorMessage=" Первое слагаемое должно иметь вид числа в диапазоне от 1 до 100"**>  
 <**f:validateLongRange minimum="1" maximum="100"** />  
 </**h:inputText**>  
 <**h:commandButton value="+" action="#{client.sum(client.slag1, client.slag2)}"**/>  
 <**h:commandButton value="-" action="#{client.dif(client.slag1, client.slag2)}"**/>  
 <**h:commandButton value="\*" action="#{client.mult(client.slag1, client.slag2)}"**/>  
 <**h:commandButton value="/" action="#{client.sub(client.slag1, client.slag2)}"**/>  
 <**h:inputText value="#{client.slag2}" validatorMessage=" Второе слагаемое должно иметь вид числа в диапазоне от 1 до 100"**>  
 <**f:validateLongRange minimum="1" maximum="100"** />  
 </**h:inputText**>  
 <**h:outputText value="="**/>  
 <**h:outputText value="#{client.result}"** />   
 </**h:panelGrid**>  
 </**h:form**>  
   
 <**h:panelGrid columns="2" bgcolor="#D6A6F3" width="600" style="font-size**: 20**px**;**"**>  
 <**h:link value="Вернутся к проверке пароля" style="color**: **black" outcome="index"** />  
 </**h:panelGrid**>  
 </**h:body**>  
</**html**>

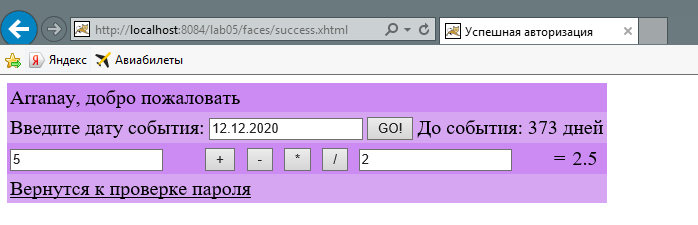


Рисунок 3. Страница после ввода правильного пароля.

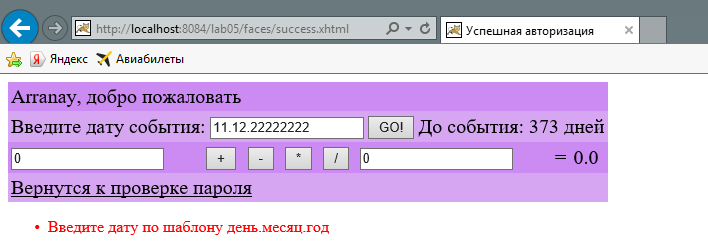


Рисунок 4. Валидация на странице.

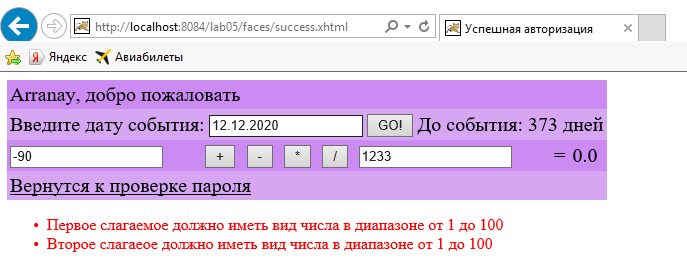


Рисунок 5. Валидация на странице.

*<?***xml version='1.0' encoding='UTF-8'** *?>***<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"*>***<**html xmlns="http://www.w3.org/1999/xhtml"  
 xmlns:h="http://xmlns.jcp.org/jsf/html"**>  
 <**h:head**>  
 <**title**>Ошибка</**title**>  
 </**h:head**>  
 <**h:body**>  
 <**h:panelGrid columns="2" bgcolor="#FF8A8D" width="300" style="font-size**: 20**px**;**"**>  
 <**h:outputText value="Неверный пароль."** />  
 <**h:outputText value="(」°ロ°)」"** />  
 <**h:link value="Назад" style="color**: **black" outcome="index"** />  
 </**h:panelGrid**>  
 </**h:body**>  
</**html**>

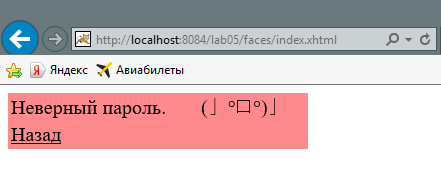


Рисунок 6. Страница при вводе неверного пароля.

ВЫВОД

В ходе лабораторной работы были изучены базовые возможности каркаса JavaServer Faces для разработки веб-приложений (организация интерфейса пользователя, обработка событий, навигация) и получены навыки реализации веб-приложений с использованием данного каркаса.