Student ID__________________Student Name____________________________________

Web Applications Architecture and Frameworks DE

Midterm Exam February 27, 2016

PRIVATE AND CONFIDENTIAL

1. **Allotted exam duration is 2 hours.**
2. **Closed book/notes.**
3. **No personal items including electronic devices (cell phones, computers, calculators, PDAs).**
4. **Cell phones must be turned in to your proctor before beginning exam.**
5. **No additional papers are allowed. Sufficient blank paper is included in the exam packet.**
6. **Exams are copyrighted and may not be copied or transferred.**
7. **Restroom and other personal breaks are not permitted.**
8. **Total exam including questions and scratch paper must be returned to the proctor.**

8 blank pages are provided for writing the solutions and/or scratch paper. All 8 pages must be handed in with the exam

BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTED TIME IS GIVEN FOR EVERY QUESTION.

Write your name and student id at the top of this page.

Write clearly. If I cannot read it, I cannot give any points.

Question 1: [10 points] {10 minutes}

Explain the difference between a JSF action event and a JSF actionlistener event

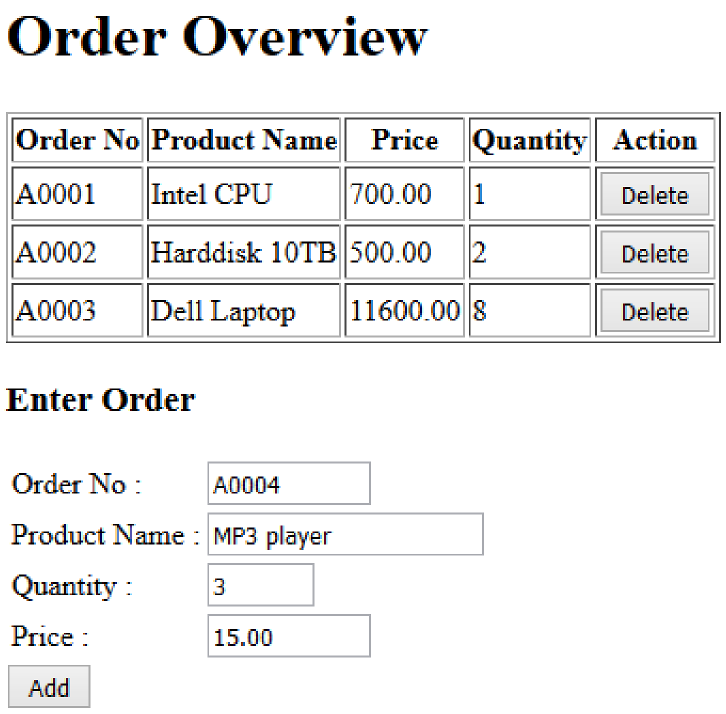
Your answer:

JSF action:
Can be used for navigation

JSF actionlistener:
The corresponding method gets the event object as parameter
Cannot be used for navigation

Question 2: [25 points] {40 minutes}

Write a JSF application with the following behavior:


Order Overview

Order No	Product Name	Price	Quantity	Action
A0001	Intel CPU	700.00	1	Delete
A0002	Harddisk 10TB	500.00	2	Delete
A0003	Dell Laptop	11600.00	8	Delete

Enter Order

Order No :	A0004
Product Name :	MP3 player
Quantity :	3
Price :	15.00

The application consists of 1 JSF page that allows you to **add** and **remove** orderlines to and from an Order. Clicking the **Delete** button should remove the row from the table. Clicking the **Add** button should add a new row to the table with the corresponding orderline.

Your implementation should follow the following requirements:

1. The application should follow the correct **Model-View-Controller** principles using JSF.
2. You are **not** allowed to use HTML, JSP's or servlets.
3. For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code**.
4. **Do NOT write getter and setter methods!**
5. **For web pages, you only need to write the code between the <body> and </body> tags. Do not write the code for namespaces**

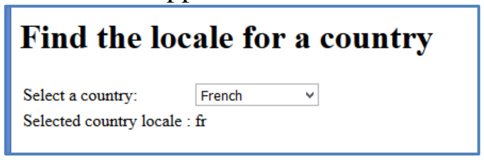
```
public class Order {
  String orderNo;
  String productName;
  BigDecimal price;
  int qty;
```

```
@ManagedBean(name="order")
@SessionScoped
public class OrderBean implements Serializable{

	String orderNo;
	String productName;
	BigDecimal price;
	int qty;

	private static final ArrayList<Order> orderList =
		new ArrayList<Order>(Arrays.asList(

		new Order("A0001", "Intel CPU",
				new BigDecimal("700.00"), 1),
		new Order("A0002", "Harddisk 10TB",
				new BigDecimal("500.00"), 2),
		new Order("A0003", "Dell Laptop",
				new BigDecimal("11600.00"), 8),
		new Order("A0004", "Samsung LCD",
				new BigDecimal("5200.00"), 3),
		new Order("A0005", "A4Tech Mouse",
				new BigDecimal("100.00"), 10)
	));


	public String addAction() {

		Order order = new Order(this.orderNo, this.productName,
			this.price, this.qty);

		orderList.add(order);

		return null;
	}

	public String deleteAction(Order order) {

		orderList.remove(order);
		return null;
	}

}
```

```
<h:body>
  <h1>Order Overview</h1>
  <h:form>
    <h:dataTable value="#{order.orderList}" var="o" border="1">
      <h:column>
        <f:facet name="header">Order No</f:facet>
        <h:outputText  value="#{o.orderNo}"/>

      </h:column>
      <h:column>
        <f:facet name="header">Product Name</f:facet>
        <h:outputText  value="#{o.productName}"/>
      </h:column>
      <h:column>
        <f:facet name="header">Price</f:facet>
        <h:outputText  value="#{o.price}"/>
      </h:column>
      <h:column>
        <f:facet name="header">Quantity</f:facet>
        <h:outputText  value="#{o.qty}"/>
      </h:column>
      <h:column>
        <f:facet name="header">Action</f:facet>
        <h:commandButton value="Delete" action="#{order.deleteAction(o)}" />

      </h:column>
    </h:dataTable>

    <h3>Enter Order</h3>
    <table>
      <tr>
        <td>Order No :</td>
        <td><h:inputText size="10" value="#{order.orderNo}" /></td>
      </tr>
      <tr>
        <td>Product Name :</td>
        <td><h:inputText size="20" value="#{order.productName}" /></td>
      </tr>
      <tr>
        <td>Quantity :</td>
        <td><h:inputText size="5" value="#{order.qty}" /></td>
      </tr>
      <tr>
        <td>Price :</td>
        <td><h:inputText size="10" value="#{order.price}" /></td>
      </tr>
    </table>

    <h:commandButton value="Add" action="#{order.addAction}" />

  </h:form>
</h:body>
```

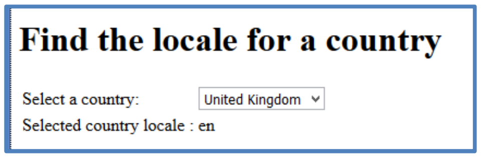
Question 3: [20 points] {30 minutes}

Write a JSF application with the following behavior:



Find the locale for a country

Select a country: French
Selected country locale : fr



Find the locale for a country

Select a country: United Kingdom
Selected country locale : en

When you select a country from the list of countries, the page shows the corresponding locale.

Your implementation should follow the following requirements:

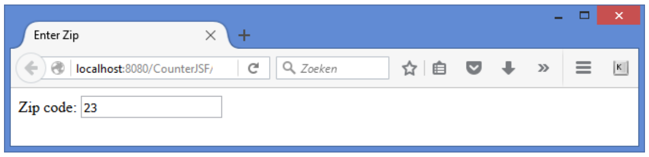
1. You are **not** allowed to use HTML, JSP's or servlets.
2. For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code**.
3. **Do NOT write getter and setter methods!**
4. **For web pages, you only need to write the code between the <body> and </body> tags. Do not write the code for namespaces**

```
<h:body>
  <h1>Find the locale for a country</h1>
    <h:form>
      <h:panelGrid columns="2">
         Select a country:
         <h:selectOneMenu value="#{country.localeCode}" onchange="submit()"
                  valueChangeListener="#{country.countryLocaleCodeChanged}">
             <f:selectItems value="#{country.countryInMap}" />
         </h:selectOneMenu>
         Selected country locale :
         <h:outputText id="country" value="#{country.localeCode}"  />
       </h:panelGrid>
    </h:form>
</h:body>
```

```java
@ManagedBean(name="country")
@RequestScoped
public class CountryBean implements Serializable{

	private static final long serialVersionUID = 1L;

	private static Map<String,String> countries;

	private String localeCode = "en"; //default value

	public CountryBean(){
		countries = new HashMap<String,String>();
		countries.put("United Kingdom", "en"); //label, value
		countries.put("French", "fr");
		countries.put("German", "de");
		countries.put("USA", "us");
	}

	public void countryLocaleCodeChanged(ValueChangeEvent e){
		//assign new value to localeCode
		localeCode = e.getNewValue().toString();

	}
```

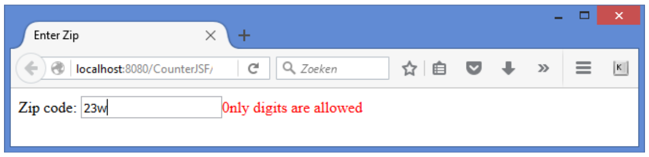
Question 4: [20 points] {20 minutes}

Write the following application with JSF:

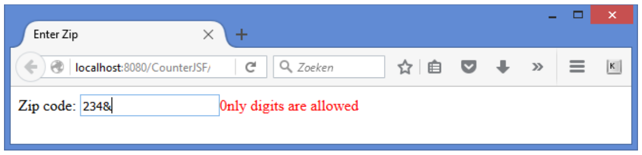


Zip code: 23

In the Zip code field you enter a zip code that consists of 5 digits. The application allows you to enter digits in the Zip code field. The moment you type something else that is not a digit, you get the error message: "**Only digits are allowed**"

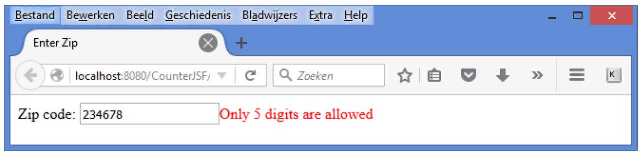


Zip code: 23w Only digits are allowed



Zip code: 2348 Only digits are allowed

If you type more than 5 digits, you get the error message: "**Only 5 digits are allowed**"



Zip code: 234678 Only 5 digits are allowed

Notice that these error messages appear while you are typing. This JSF page does not have a button.

Your implementation should follow the following requirements:

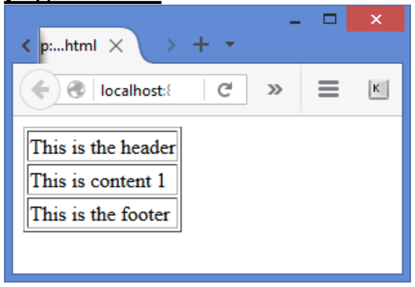
1. You are **not** allowed to use HTML, JSP's or servlets.
2. For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code**.
3. **Do NOT write getter and setter methods!**
4. **For web pages, you only need to write the code between the <body> and </body> tags. Do not write the code for namespaces**

```
  <h:body>
    <h:form>
      Zip code:
      <h:inputText id="zipcode" value="#{zipcodeBean.zipcode}"
        validator="#{zipcodeBean.validateZip}"
        validatorMessage="Only digits are allowed">
             >
        <f:validateLongRange maximum="99999" minimum="0"/>
        <f:ajax event="keyup"  render="ziperror"/>
      </h:inputText>
      <h:message id="ziperror" for="zipcode" style="color: red" />
    </h:form>
  </h:body>


@ManagedBean
@RequestScoped
public class ZipcodeBean {

  private String zipcode;

  public void validateZip(FacesContext context, UIComponent toValidate, Object value){
   String zipcode = (String) value;

   if (zipcode.length()>5){
     FacesMessage message = new FacesMessage("Only 5 digits are allowed");
    context.addMessage(toValidate.getClientId(context), message);
   }
  }

}
```

Question 5: [20 points] {20 minutes}

Write the following application with JSF and facelets:

page1.xhtml

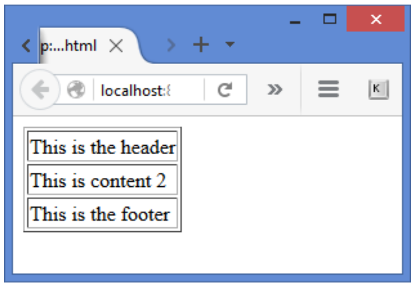


This is the header

This is content 1

This is the footer

page2.xhtml



This is the header

This is content 2

This is the footer

Every page we have to make has the same header and footer. We need to use facelets to make it easy to give every page the same header and footer.
Given are the following files:

header.xhtml:
This is the header

footer.xhtml:
This is the footer

content1.xhtml:
This is content1

content2.xhtml:
This is content2

Write all the necessary files that are missing.

Your implementation should follow the following requirements:

1. You are **not** allowed to use HTML, JSP's or servlets.
2. You need to use **facelets**
3. For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code**.
4. **Do NOT write getter and setter methods!**
5. **For web pages, you only need to write the code between the <body> and </body> tags. Do not write the code for namespaces**

template.xhtml
```
 <h:body>
   <h:panelGrid columns="1" border="1">
    <ui:insert name="header" />
    <ui:insert name="content" />
    <ui:insert name="footer" />
  </h:panelGrid>
 </h:body>
```

page1.xhtml
```
 <h:body>
  <ui:composition template="template.xhtml">
   <ui:define name="header">
    <ui:include src="header.xhtml" />
   </ui:define>

   <ui:define name="content">
    <ui:include src="content1.xhtml" />
   </ui:define>

   <ui:define name="footer">
    <ui:include src="footer.xhtml" />
   </ui:define>
  </ui:composition>
 </h:body>
```

page2.xhtml
```
 <h:body>
  <ui:composition template="template.xhtml">
   <ui:define name="header">
    <ui:include src="header.xhtml" />
   </ui:define>

   <ui:define name="content">
    <ui:include src="content2.xhtml" />
   </ui:define>

   <ui:define name="footer">
    <ui:include src="footer.xhtml" />
   </ui:define>
  </ui:composition>
 </h:body>
```