Student ID___________________Student Name______________________________________

Web Applications Architecture and Frameworks DE

Final Exam February 4 2017

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.**

3 blank pages are provided for writing the solutions and/or scratch paper. All 3 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.

Question 1 [10 points] {15 minutes}

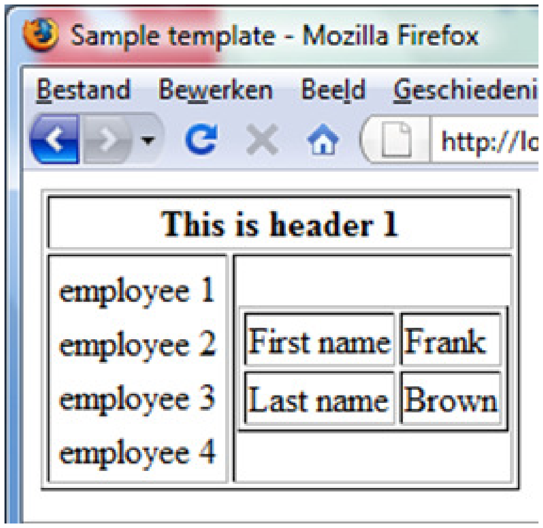
JSF 2.0 supports different scopes for managed beans.
Explain clearly the different scopes that JSF 2.0 supports and explain shortly and clearly what they mean**.**

@RequestScoped: a bean in this scope lives as long as the HTTP request-response lives
@ViewScoped: a bean in this scope lives as long as you're interacting with the same JSF view in the browser window/tab
@SessionScoped: a bean in this scope lives as long as the HTTP session lives
@ApplicationScoped: a bean in this scope lives as long as the web application lives
…

Question 2. [20 points] {20 minutes}

We need to write the following application using facelets:

We have 2 pages: **main.xhtml** and **main2.xhtml**

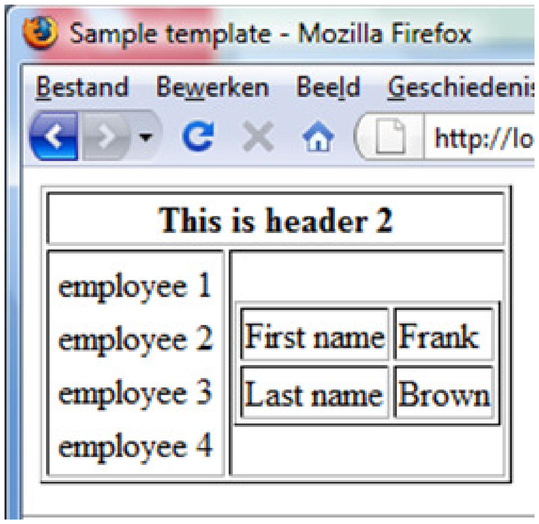


This is header 1

employee 1
employee 2
employee 3
employee 4

First name	Frank
Last name	Brown

main.xhtml shows a header at the top, a menu at the left side and content at the remainder of the page.



This is header 2

employee 1
employee 2
employee 3
employee 4

First name	Frank
Last name	Brown

main2.xhtml has the same structure as main.xhtml, only the header is different.

We have the following JSF pages:

header.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition>
    <h:outputText value="This is header 1" />
  </ui:composition>
</html>
```

header2.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition>
    <h:outputText value="This is header 2" />
  </ui:composition>
</html>
```

menu.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition>
    <h:panelGrid columns="1">
    <h:outputText value="employee 1" />
    <h:outputText value="employee 2" />
    <h:outputText value="employee 3" />
    <h:outputText value="employee 4" />
    </h:panelGrid>
  </ui:composition>
</html>
```

content.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition>
    <h:panelGrid columns="2" border="1">
      <h:outputText value="First name" />
      <h:outputText value="Frank" />
      <h:outputText value="Last  name" />
      <h:outputText value="Brown" />
    </h:panelGrid>
  </ui:composition>
</html>
```

Complete the partial given code of the other necessary JSF pages such that the application works with the given behavior using facelets.

template.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">
  <head>
    <title>Sample template</title>
  </head>
  <body>
    <h:panelGrid columns="2" border="1">

        <f:facet name="header">
            <ui:insert name="header" > Default Header </ui:insert>
       </f:facet>
      <ui:insert name="menu" > Default Menu </ui:insert>
      <ui:insert name="content" > Default Content </ui:insert>
    </h:panelGrid>
  </body>
</html>
```

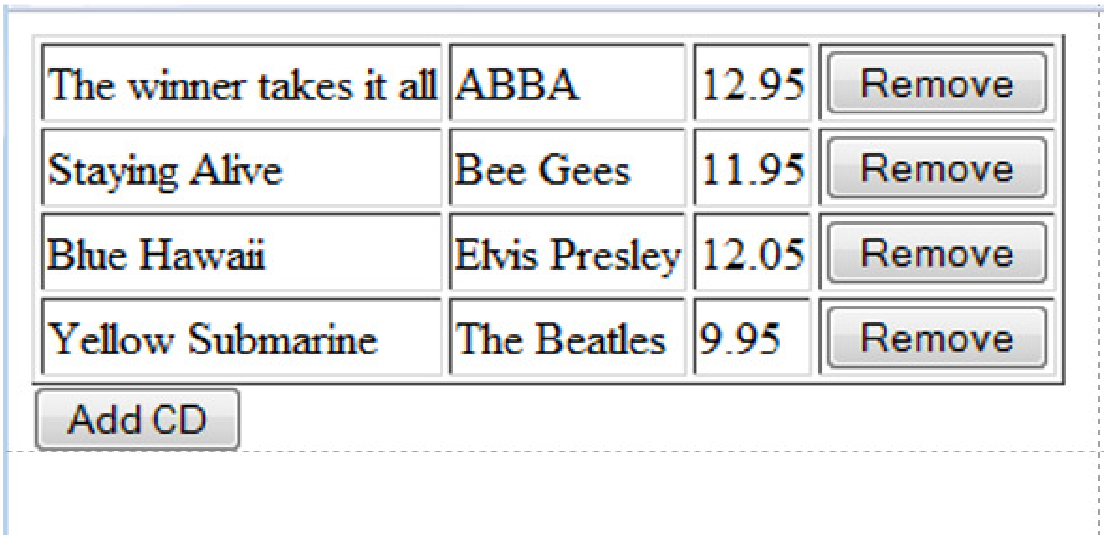
main.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header.xhtml" />
    </ui:define>
    <ui:define name="menu">
      <ui:include src="menu.xhtml" />
    </ui:define>
    <ui:define name="content">
      <ui:include src="content.xhtml" />
    </ui:define>
  </ui:composition>
 </html>
```

main2.xhtml

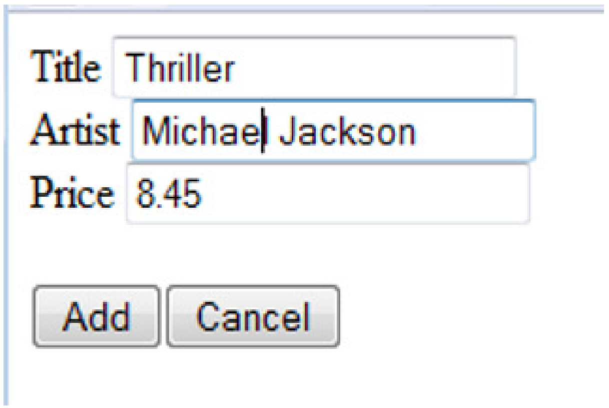
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core" xml:lang="en" lang="en">

  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header2.xhtml" />
    </ui:define>
    <ui:define name="menu">
      <ui:include src="menu.xhtml" />
    </ui:define>
    <ui:define name="content">
      <ui:include src="content.xhtml" />
    </ui:define>
  </ui:composition>
</html>
```

Question 3 CD-Application [30 points] {30 minutes}
Write a JSF application with the following behavior:



The winner takes it all	ABBA	12.95	Remove
Staying Alive	Bee Gees	11.95	Remove
Blue Hawaii	Elvis Presley	12.05	Remove
Yellow Submarine	The Beatles	9.95	Remove

The application shows a list of CD's in a table. If you click the Remove button, that particular CD will be removed from the list.
If you click the Add CD button, then this page will be shown:



Title Thriller
Artist Michael Jackson
Price 8.45

When you enter the CD information, and you click the Add button, then you return to the previous CD table page, and you see that this new CD is added to the list of CD's.

Complete the partial given code such that the application works with the given behavior. You have to implement all necessary java code, required annotations and JSF tags that are missing. Write the application according the best practices we studied in the course. IMPORTANT: do not write getter and setter methods!

<u>cdcollection.xhtml</u>

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>CD collection</title>
    </h:head>
    <h:body>
    <h:form>
     <h:dataTable value="#{ cdtable.cdlist.cdlist}" var="cd" border="1" >
        <h:column>
          <h:outputText value="#{cd.title}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{cd.artist}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{cd.price}"/>
        </h:column>
        <h:column>
           <h:commandButton value="Remove" action="#{ cdtable.removeCD(cd)}"/>
        </h:column>
        </h:dataTable>
        <h:commandButton value="Add CD"
                     action="addcd"/>
    </h:form>
    </h:body>
</html>
```

<u>addcd.xhtml</u>

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
     <title>Add new CD</title>
  </h:head>
  <h:body>
    <h:form>
    Title
    <h:inputText value="#{addcd.title}"/>
    <br />
    Artist
    <h:inputText value="#{addcd.artist}"/>
    <br />
    Price
    <h:inputText value="#{addcd.price}"/>
    <br />
    <br />
    <h:commandButton value="Add"
               action="#{addcd.add}"/>
    <h:commandButton value="Cancel" type="submit"
               action="cdcollection"/>
  </h:form>
  </h:body>
</html>
```

<u>CD.java</u>

```
public class CD {
  private String title;
  private String artist;
  private double price;

   //getters and setters are not shown
}
```

Cdtable.java

```
@ManagedBean
@RequestScoped
public class Cdtable {


  @ManagedProperty(value="#{ cDCollection }")
   CDCollection cdlist;


   public String removeCD(CD cd) {
     cdlist.removeCD(cd.getTitle());
     return "";
   }

   //getters and setters are not shown (do not write getter and setter methods)
}
```

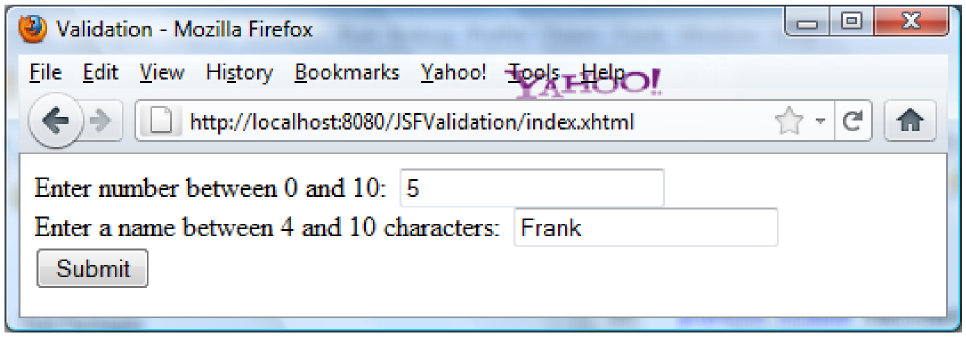
Addcd.java

```
@ManagedBean
@RequestScoped
public class Addcd {


   private String title = "";
   private String artist = "";
   private double price ;

   @ManagedProperty(value="#{ cDCollection }")
   CDCollection cdlist;

   public String add() {
      CD newCD = new CD(title, artist,price);
      cdlist.addCD(newCD);
      return "cdcollection";
   }


     //getters and setters are not shown (do not write getter and setter methods)
}
```

```
@ManagedBean
@SessionScoped
public class CDCollection implements Serializable{
   private Collection<CD> cdlist = new ArrayList();

   public CDCollection() {
      cdlist.add(new CD("The winner takes it all ", "ABBA", 12.95));
      cdlist.add(new CD("Staying Alive ", "Bee Gees", 11.95));
      cdlist.add(new CD("Blue Hawaii ", "Elvis Presley", 12.05));
      cdlist.add(new CD("Yellow Submarine ", "The Beatles", 9.95));

   }

   public void addCD(CD cd) {
      cdlist.add(cd);
   }

   public void removeCD(String title) {
      Iterator iter = cdlist.iterator();
      while (iter.hasNext()) {
         CD cd = (CD) iter.next();
         if (cd.getTitle().equals(title)) {
            iter.remove();
         }
      }
   }



}
```

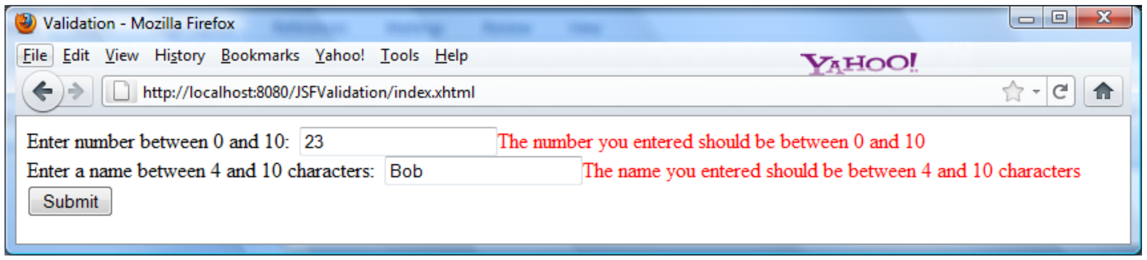
Question 4 Validation [15 points] {15 minutes}

Write an JSF application with the following validation behavior:


Validation - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://localhost:8080/JSFValidation/index.xhtml

Enter number between 0 and 10: 5
Enter a name between 4 and 10 characters: Frank
Submit

When the user enters a number that is not between 0 and 10, and/or the user enters a name that is not between 4 and 10 characters long, then we get the following result:


Validation - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://localhost:8080/JSFValidation/index.xhtml

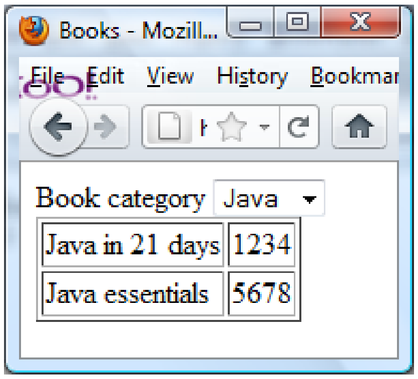
Enter number between 0 and 10: 23 The number you entered should be between 0 and 10
Enter a name between 4 and 10 characters: Bob The name you entered should be between 4 and 10 characters
Submit

Complete the partial given code such that the application works with the given behavior. **You only have to complete the code for the xhtml file.**

```
@ManagedBean
@RequestScoped
public class ValidationBackingBean {

  private int number;
  private String name;
  private double doublenumber;
  private String phone;

    public String save(){
        return null;
    }

   // getter and setter methods are not shown
}
```

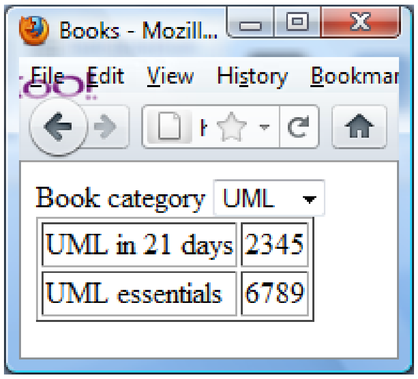
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>Validation</title>
    </h:head>
    <h:body>
        <h:form>
            Enter number between 0 and 10:
            <h:inputText id="inputNumber" value="#{validationBackingBean.number}"
                         validatorMessage="The number you entered should be between 0 and 10">
                <f:validateLongRange maximum="10" minimum="0" />
            </h:inputText>
            <h:message style="color: red" for="inputNumber"/>
            <br />
            Enter a name between 4 and 10 characters:
            <h:inputText id="inputName" value="#{validationBackingBean.name}"
                         validatorMessage="The name you entered should be between 4 and 10
characters">
                <f:validateLength maximum="10" minimum="4"/>
            </h:inputText>
            <h:message style="color: red" for="inputName"/>
            <br />
            <h:commandButton value="Submit" action="#{validationBackingBean.save}"/>
        </h:form>
    </h:body>
</html>
```

Question 5 Events [20 points] {20 minutes}



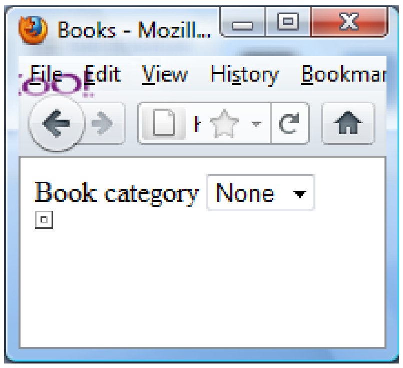
Book category Java

Java in 21 days	1234
Java essentials	5678



Book category UML

UML in 21 days	2345
UML essentials	6789



Book category None

Write an JSF application with the following behavior:
The application shows a selectOneMenu control with the values Java, UML and None.
If you select Java, then you see a table showing 2 Java books.
If you select UML, then you see a table showing 2 UML books.
If you select None, then the table is empty.
The first column in the table is the name of the book, and the second column is the ISBN number (in this example just a string containing 4 characters)

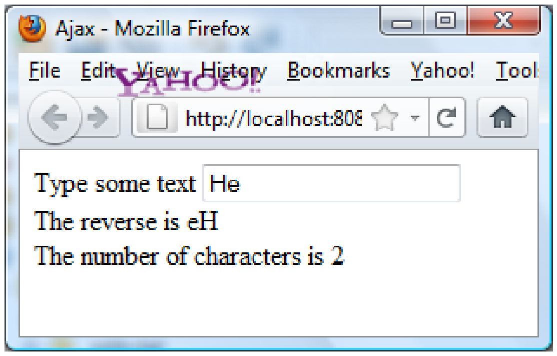
Complete the partial given code such that the application works with the given behavior. You only have to complete the code for the xhtml file and the managed bean.
IMPORTANT: do not write getter and setter methods!

books.xhtml

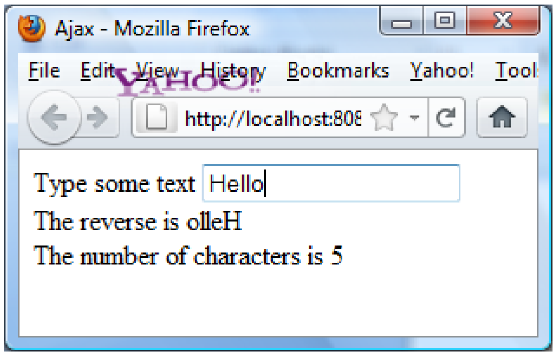
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>Books</title>
    </h:head>
    <h:body>
        <h:form>
    Book category
    <h:selectOneMenu value="#{bookbean.selectedBook}"
                     valueChangeListener="#{bookbean.changeBook}"
                     onchange="submit()">
        <f:selectItems value="#{bookbean.books}"/>
    </h:selectOneMenu>
    <br />
    <h:dataTable value="#{bookbean.booklist}" var="book" border="1" >
        <h:column>
          <h:outputText value="#{book.name}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{book.isbn}"/>
        </h:column>
        </h:dataTable>
        </h:form>
    </h:body>
```

```
import java.util.*;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.event.ValueChangeEvent;
import javax.faces.model.SelectItem;

@ManagedBean
@RequestScoped
public class Bookbean {


   private String selectedBook;
   private Collection<SelectItem> books = new ArrayList();
   private Collection<Book> booklist= new ArrayList<Book>();

   public Bookbean(){
      books.add(new SelectItem("None","None"));
      books.add(new SelectItem("Java","Java"));
      books.add(new SelectItem("UML","UML"));
   }

   public void changeBook(ValueChangeEvent valueChangeEvent) {
      if (valueChangeEvent.getNewValue().toString().equals("Java")){
         booklist.add(new Book("Java in 21 days", "1234"));
         booklist.add(new Book("Java essentials", "5678"));
      } else if (valueChangeEvent.getNewValue().toString().equals("UML")){
         booklist.add(new Book("UML in 21 days", "2345"));
         booklist.add(new Book("UML essentials", "6789"));
      } else {
         booklist.clear();
      }
   }




}
```

Question 6 AJAX [15 points] {20 minutes}



Type some text He
The reverse is eH
The number of characters is 2



Type some text Hello
The reverse is olleH
The number of characters is 5

Write an JSF application with the following AJAX behavior:
Whenever you type something in the inputtext control, an request is send to the server, and the reverse of the entered text, and the number of characters entered is shown on the page. This behavior should be implemented with AJAX, such that only the reverse text and the number of characters are rendered.

You can get the reverse of a String with the following code:

reverse= new StringBuffer(text).reverse().toString();

were the String reverse is the reverse of the String text.

Complete the partial given code such that the application works with the given behavior. You only have to complete the code for the xhtml file and the managed bean.
IMPORTANT: do not write getter and setter methods!

ajax.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
   <h:head><title>Ajax</title></h:head>
   <h:body>
      <h:form>
         Type some text
         <h:inputText value="#{bean.text}" >
            <f:ajax event="keyup" render="reverse count" listener="#{bean.countListener}"/>
         </h:inputText>
         <br />
         The reverse is <h:outputText id="reverse" value="#{bean.reverse}" />
         <br />
         The number of characters is <h:outputText id="count" value="#{bean.count}" />
      </h:form>
   </h:body>
</html>
```

```
@ManagedBean
@RequestScoped
public class Bean {



    private String text;
    private String reverse;
    private int count;

    public void countListener(AjaxBehaviorEvent event) {
        reverse= new StringBuffer(text).reverse().toString();
        count = text.length();
    }










}
```