Student ID__________________Student Name______________________________________

Web Applications Architecture and Frameworks DE

Midterm Exam June 27, 2015

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.**

2 blank pages are provided for writing the solutions and/or scratch paper. All 2 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}

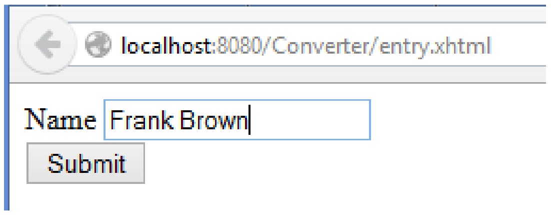
Give the 6 phases of the JSF lifecycle and describe in one or two sentences what specific task is done in this phase

Phase	**Specific task of this phase**
Restore view	**Create/retrieve component tree**
Apply request	**The values of the components in the component tree are updated with the values from the request**
Process validations	**Validate the values in the component tree**
Update model values	**Find the backingbean and update the values in the backing bean**
Invoke application	**Call the registered action listeners**
Render response	**Create the Http response**

Question 2: [25 points] {25 minutes}

Write a JSF application with the following behavior:
We have 2 pages: entry.xhtml and result.xhtml:

When you request the entry page, you get the following page:

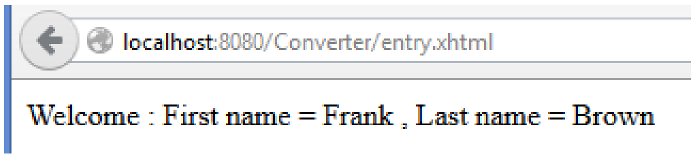


localhost:8080/Converter/entry.xhtml

Name Frank Brown

Submit

When you enter your full name (first name followed by a space followed by lastname) and you click the **Submit** button, then the following page (result.xhtml) will be shown:



localhost:8080/Converter/entry.xhtml

Welcome : First name = Frank , Last name = Brown

Complete the partial given code such that the application works with the given behavior. You have to implement all java code, required annotations and JSF tags that are missing. For this exercise you have to write a custom converter. You are not allowed to change the given code.
IMPORTANT: do not write getter and setter methods!

<u>entry.xhtml</u>

```
<html ...>
  <h:head>
    <title>Entry</title>
  </h:head>
  <h:body>
    <h:form>
      Name
      <h:inputText value="#{personBean.person}">
        <f:converter converterId="personConverter"/>

      </h:inputText>
      <br />
      <h:commandButton value="Submit"
                action="result.xhtml"/>

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

result.xhtml

```
<html ...>
  <h:head>
    <title>Result</title>
  </h:head>
  <h:body>
    Welcome :
    <h:outputText value="#{personBean.person}">
      <f:converter converterId="personConverter"/>
    </h:outputText>
  </h:body>
</html>
```

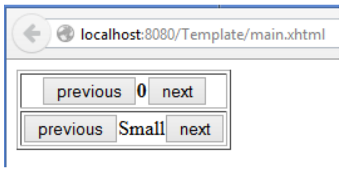
```
class Person {
  private String firstname;
  private String lastName;

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

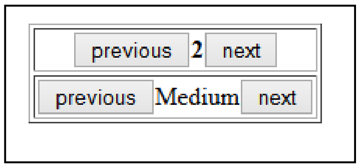
```
@ManagedBean
@RequestScoped
public class PersonBean {

  Person person ;

  public String submit(){
    return "";
  }
 // getters and setters are not shown
}
```

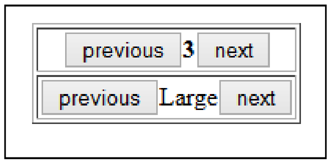
```
@FacesConverter( value="personConverter" )
public class PersonConverter  implements Converter {
  public Object getAsObject(FacesContext context, UIComponent component, String value) {
    if ("".equals(value)){
      return null;
    }
    String firstname = value.substring(0,value.indexOf(' '));
    String lastname = value.substring(value.indexOf(' ')+1,value.length());
   return new Person(firstname, lastname);
  }
  public String getAsString(FacesContext context, UIComponent component, Object value) {
      Person person = (Person)value;
      return "First name = "+person.getFirstname() +" , Last name = "+ person.getLastName();
  }
}
```

Question 3 [30 points] {40 minutes}

Write a JSF application with the following behavior:
The application has 2 pages: main.xhtml and main2.xhtml

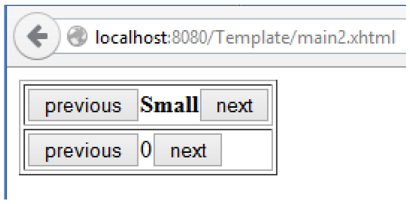


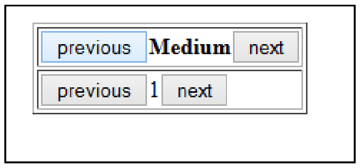


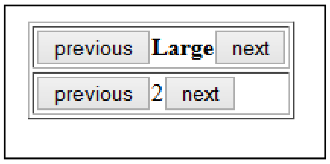


When you request **main.xhtml**, you get a page with 2 sections.
The top section contains a counter that counts up or down. The counter starts at value 0, and when you click the next button, the counter value is 1, and when you click the next button again, the counter value is 2. The previous button works in the opposite direction.

The bottom section shows the sizes “Small”, “Medium” and “Large”.
The size starts at value “Small”, and when you click the next button, the value is “Medium”, and when you click the next button again, the counter value is “Large”. When you click the next button again, the value returns back to “Small”. The previous button works in the opposite direction.







When you request **main2.xhtml**, you also get a page with 2 sections, but now the counter and size sections are switched. The size is shown at the top section, and the counter at the bottom section. The counter and the size works exactly the same as explained for main.xhtml.

Write the code for this application using **facelets**.

Write all necessary code such that the application works with the requested behavior.
You have to implement all java code, required annotations and JSF pages.
This application does not have a facelets.xml, so you have to use annotations for JSF configuration.
IMPORTANT:

- **Do not write namespaces in the xhtml pages!**
- **Do not write getter and setter methods in the java classes!**

Template.xhtml

```
<html …>
 <body>
  <h:panelGrid columns="2" border="1">
   <f:facet name="header">
    <ui:insert name="header" />
   </f:facet>
   <ui:insert name="body" />
  </h:panelGrid>
 </body>
</html>
```

Counter.xhtml

```
<html …>
 <ui:composition>
    <h:form>
         <h:commandButton action="#{counter.decrement}" value="previous" />
         <h:outputText id="language" value="#{counter.value}" />
         <h:commandButton action="#{counter.increment}" value="next" />
   </h:form>
 </ui:composition>
</html>
```

size.xhtml

```
<html …>
  <ui:composition>
    <h:form>
      <h:commandButton action="#{sizeBean.next}" value="previous" />
      <h:outputText value="#{sizeBean.size}" />
      <h:commandButton action="#{sizeBean.previous}" value="next" />
    </h:form>
  </ui:composition>
</html>
```

main.xhtml

```
<html …>
  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="counter.xhtml" />
    </ui:define>
    <ui:define name="body">
      <ui:include src="size.xhtml" />
    </ui:define>
  </ui:composition>
</html>
```

main2.xhtml

```
<html ...>
  <ui:composition template="template.xhtml">
      <ui:define name="header">
         <ui:include src="size.xhtml" />
      </ui:define>
      <ui:define name="body">
         <ui:include src="counter.xhtml" />
      </ui:define>
   </ui:composition>
</html>
```

```
@ManagedBean
@SessionScoped
public class Counter {

   int value=0;

   public String increment(){
      value++;
      return"";
   }

      public String decrement(){
      value--;
      return"";
   }
}
```

```
@ManagedBean
@SessionScoped
public class SizeBean {

   private String size="Small";

   public String next() {
      if (size.equals("Small")) {
         size = "Medium";
      } else {
         if (size.equals("Medium")) {
            size = "Large";
         } else {
            size = "Small";
         }
      }
      return"";
   }

   public String previous() {
      if (size.equals("Large")) {
         size = "Medium";
      } else {
         if (size.equals("Medium")) {
            size = "Small";
         } else {
            size = "Large";
         }
      }
      return"";
   }
}
```

Question 4 [15 points] {15 minutes}

Suppose you run a JSF application that shows you the login page. The URL in the browser is http://localhost:8080/MyApp/login.xhtml.
The login.xhtml page has a button which looks like this:

<h:commandButton label="Login" action="welcome"/>

If you click the button, you will see the welcome.xhtml page.
What is the URL we see in the browser?

Your answer: http://localhost:8080/MyApp/login.xhtml

Now if the button would look like this:

<h:commandButton label="Login" action="welcome?faces-redirect=true"/>

What would be the URL we see in the browser:

Your answer: http://localhost:8080/MyApp/welcome.xhtml

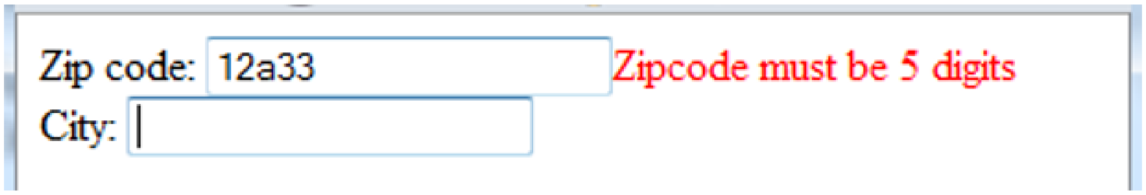
Explain the difference between the two different options given above.

Your answer:

The first option does forward navigation and teh second option does a redirect with an extra rountrip between client and server

Question 5 [15 points] {15 minutes}

Implement the following JSF application:



Zip code: 12a33 Zipcode must be 5 digits

City:

We have one page with 2 input fields. First you enter a zip code which does not consist of 5 digits. Then you select the input field for the City, and then immediately you see the validation error. So when you remove the focus of the zip code field, the zip field will be validated.

Complete the partial given code such that the application works with the given behavior.

```
@ManagedBean
@RequestScoped
public class ZipcodeBean {
  private String zipcode;
  private String city;

  public void validateZip(FacesContext context, UIComponent toValidate, Object value){
    String zipcode = (String) value;
    Pattern mask = null;
    mask = Pattern.compile("[0-9]{5}");
    /* Check to see if the value is a zip code */
    Matcher matcher = mask.matcher(zipcode);
    if (!matcher.matches()){
      FacesMessage message = new FacesMessage("Zipcode must be 5 digits");
      context.addMessage(toValidate.getClientId(context), message);
    }
  }
 ...
}
```

entry.xhtml:

```
<html …>
<body>
<h:form>
        Zip code:
        <h:inputText id="zipcode" value="#{zipcodeBean.zipcode}"
                validator="#{zipcodeBean.validateZip}"
                required="true"
                requiredMessage="Zip code is required"
                >
          <f:ajax event="blur"  render="ziperror"/>
        </h:inputText>
        <h:message id="ziperror" for="zipcode" style="color: red" />
        <br />
        City:
        <h:inputText value="#{zipcodeBean.city}" />
    </h:form>
</body>
</html>
```

Question 6. SCI [5 points] {10 minutes}

Describe how we can relate the concept of **JSF facelets** to the principles of SCI. Your answer should be about half a page, but should not exceed one page (handwritten). The number of points you get for this question depend on how well you explain the relationship between **JSF facelets** and the principles of SCI.

Your answer: