

Student ID \_\_\_\_\_ Student Name \_\_\_\_\_

**CS545 Web Applications Architecture and Frameworks**  
**DE Final Exam**  
**March 1, 2014**

PRIVATE AND CONFIDENTIAL

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

5 blank pages are provided for writing the solutions and/or scratch paper. All 5 pages must be handed in with the exam

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

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

**Question 1 [ 15 points ] {20 minutes}**

- a) Explain clearly the difference between action, actionlistener and valuechangelistener attributes. When would you use which attribute?
- Action: when you want to execute some code when you click a button or a hyperlink.  
Action is also used for navigation.
- Actionlistener: when you want to execute some code when you click a button or a hyperlink. You also have access to the action event, so you can use send parameters.  
Cannot be used for navigation.
- Valuechangelistener: : when you want to execute some code when the value of a control changes.
- b) Explain clearly what redirection means with respect to JSF navigation. Give the advantages and disadvantages of redirection in JSF.

With redirection, the server send a response to the browser to change the URL. The Browser changes the URL and sends a new request to the server.

Advantage: the URL is always correct.

Disadvantage: extra roundtrip and we loose the original request values.

- c) We learned that all HTTP requests for a JSF application are handled by the FacesServlet. This FacesServlet will start a sequence of steps that are executed so that everything works correctly. We call this the JSF lifecycle. Explain clearly the different steps that are started by the FacesServlet, and explain the work that is done in every step.

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 [ 40 points ] {40 minutes}

Write a JSF application with the following behavior:



The screenshot shows a web browser window titled "CD Application". The page has a header "CD list". Below the header is a table with four rows of CD information. Each row has a "Remove" button. Below the table is an "Add CD" button. At the bottom of the page, it says "Created by Frank Brown".

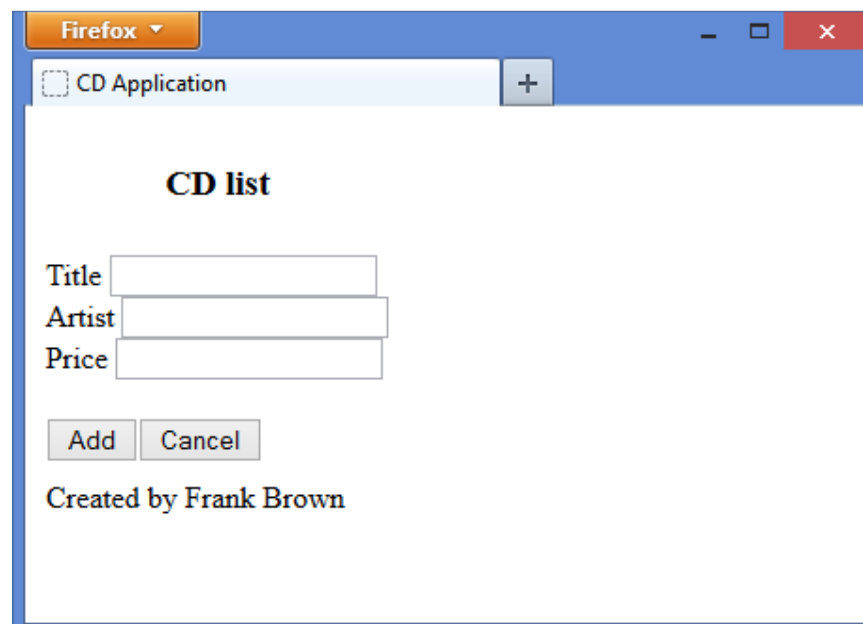
The winner takes it all	ABBA	12.95	Remove
Staying Alive	Bee Gees	11.95	Remove
Blue Hawaii	Elvis Presley	12.05	Remove
Yellow Submarine	The Beatles	9.95	Remove

Add CD

Created by Frank Brown

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:



The screenshot shows a web browser window titled "CD Application". The page has a header "CD list". Below the header are three input fields labeled "Title", "Artist", and "Price". Below the input fields are two buttons: "Add" and "Cancel". At the bottom of the page, it says "Created by Frank Brown".

Title

Artist

Price

Add Cancel

Created by Frank Brown

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

When you click the Cancel button you navigate to the previous CD table page.

The application uses a standard header and footer which is the same for all pages of the application. You have to use facelets to implement a standard header and footer for all pages.

Complete the partial given code such that the application works with the given behavior. You have to implement all java code, required annotation and JSF tags that are missing.  
This JSF application does not use a faces-config.xml file but uses annotation based configuration.  
**IMPORTANT:** do not write getter and setter methods!

#### header.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>
    <h3>CD list</h3>
  </ui:composition>

</html>
```

#### footer.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>

    Created by Frank Brown
  </ui:composition>

</html>
```

#### cd.java

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

  public CD(String title, String artist, String price) {
    this.title = title;
    this.artist = artist;
    this.price = price;
  }
  //getters and setters are not shown
}
```

## cdlistpage.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header.xhtml" />
    </ui:define>

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

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

## addcdpage.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header.xhtml" />
    </ui:define>

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

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

## cdtable.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>

    <h:form>
      <h:dataTable value="#{cdtable.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="addcdpage"/>
    </h:form>
  </ui:composition>

</html>
```

## addcdform.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>
    <h:form>
      Title
      <h:inputText value="#{cdtable.title}"/>
      <br />
      Artist
      <h:inputText value="#{cdtable.artist}"/>
      <br />
      Price
      <h:inputText value="#{cdtable.price}"/>
      <br />
      <br />
      <h:commandButton value="Add"
        action="#{cdtable.add}"/>
      <h:commandButton value="Cancel" type="submit"
        action="cdlistpage"/>
    </h:form>
  </ui:composition>
</html>
```



## template.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <h:head>
    <title>CD Application</title>
  </h:head>
  <h:body>
    <h:panelGrid columns="1" >
      <f:facet name="header">
        <ui:insert name="header" > Default Header </ui:insert>
      </f:facet>
      <ui:insert name="content" > Default Content </ui:insert>
      <f:facet name="footer">
        <ui:insert name="footer" > Default Footer </ui:insert>
      </f:facet>
    </h:panelGrid>
  </h:body>
</html>
```

## cdtable.java

**@Named**

**@SessionScoped**

```
public class Cdtable implements Serializable{
    private Collection<CD> cdlist = new ArrayList();
    private String title = "";
    private String artist = "";
    private String price = "";

    public Cdtable() {
        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"));
    }

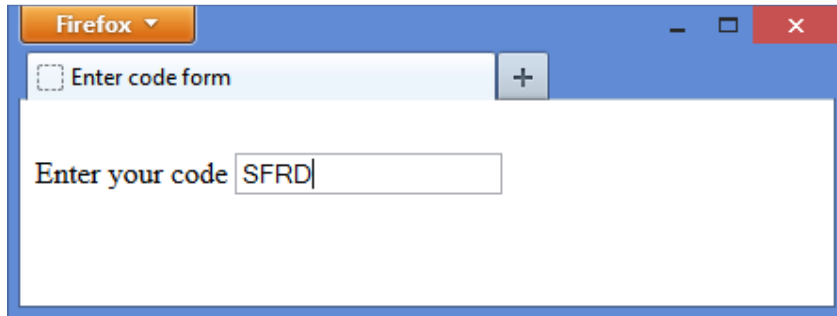
    public void removeCD(CD removeCd) {
        Iterator<CD> iter = cdlist.iterator();
        while (iter.hasNext()) {
            CD cd = iter.next();
            if (cd.getTitle().equals(removeCd.getTitle())) {
                iter.remove();
            }
        }
    }

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

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

### Question 3 [ 20 points ] {20 minutes}

We have to write the following application in JSF:



The application shows a form that asks you to enter a code.

When you start typing in the inputtext, automatically all entered characters are shown as uppercase characters. So if you type a lowercase "s", the page will show an uppercase "S".

Use **AJAX** to implement this behavior.

A String in Java has the **toUpperCase()** method to transform all character to uppercase characters.

**Complete the partial given code such that the application works with the given behavior. You have to implement all java code, required annotation and JSF tags that are missing. This JSF application does not use a faces-config.xml file but uses annotation based configuration.**

**IMPORTANT: do not write getter and setter methods!**

entercode.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
    <title>Enter code form</title>
  </h:head>
  <h:body>
    <h:form >
      <br/>
      Enter your code
      <h:inputText id="in" value="#{codebean.code}" >
        <f:ajax event="keyup" render="in"
              listener="#{codebean.update}" />
      </h:inputText>
      <br/>
    </h:form>
  </h:body>
</html>
```

```
@Named
@RequestScoped
public class Codebean {
    private String code;

    public void update(AjaxBehaviorEvent event) {
        code=code.toUpperCase();
    }

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

**Question 4 [ 20 points ] {30 minutes}**

- a. Suppose you need to validate an input field of a form to check if it is a number between 90 and 100. How would you do that? Give some example code that shows how this will work.

```
<h:inputText id="inputNumber" value="#{validationBackingBean.number}">
    <f:validateLongRange maximum="10" minimum="0"/>
</h:inputText>
```

- b. Suppose you need to validate an input field of a form to check if the entered text starts with 3 digits followed by a forward slash and then followed by 3 letters. How would you do that? Give some example code that shows how this will work.

```
<h:inputText id="phone" value="#{validationBackingBean.text}">
    <f:validateRegex pattern="([0-9]{3})/[a-z]{3}" />
</h:inputText>
```

- c. Suppose you need to convert the entered text of an input field of a form to a java.util.Date object. How would you do that? Give some example code that shows how this will work.

```
<h:inputText value="#{dateBackingBean.dateofbirth}">
    <f:convertDateTime pattern="dd/MM/yyyy"/>
</h:inputText>
```

- d. Suppose you need to convert the entered text of an input field of a form to a domain specific object, for example a phone number class. You need to use this conversion logic at a number of different places in your application. How would you do that? Give some example code that shows how this will work.

```
@FacesConverter( value="phoneConverter" )
public class Phoneconverter implements Converter {
    public Object getAsObject(FacesContext context, UIComponent component, String value) {
        if ("".equals(value)){
            return null;
        }
        PhoneNumber phone = new PhoneNumber();
        String countryCode = value.substring(value.indexOf('(')+1,value.indexOf(')'));
        String areaCode = value.substring(value.indexOf('')+1,value.indexOf('-'));
        String number = value.substring(value.indexOf('-')+1,value.length());
        phone.setCountryCode(countryCode);
        phone.setAreaCode(areaCode);
        phone.setNumber(number);
        return phone;
    }
}
```

```
public String getAsString(FacesContext context, UIComponent component, Object
value) {
    return value.toString();
}
}

<h:inputText value="#{phoneBackingBean.phonenumber}">
    <f:converter converterId="phoneConverter"/>
</h:inputText>
```

**Question 5 SCI [ 5 points ] {10 minutes}**

Describe how we can relate **JSF navigation** 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 questions depend on how well you explain the relationship between **JSF navigation** and the principles of SCI.