# EXERCISE

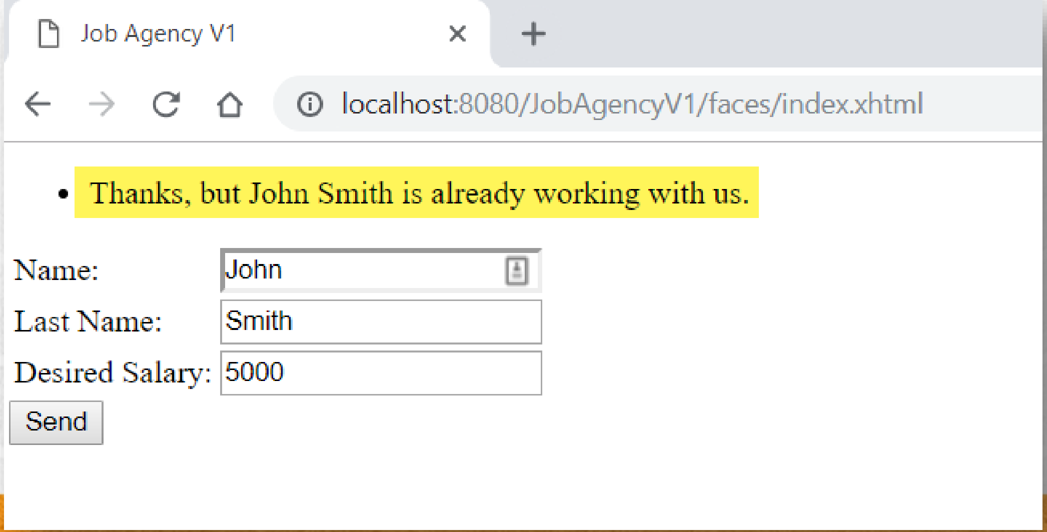
# VALIDATORS IN JSF

Global Mentoring

# EXERCISE OBJECTIVE

In this exercise we will handle the concept of Validation in JSF.

We will add the field of surname and salary desired in order to make them required fields. Part of the result should be similar to the one shown below:



- Thanks, but John Smith is already working with us.

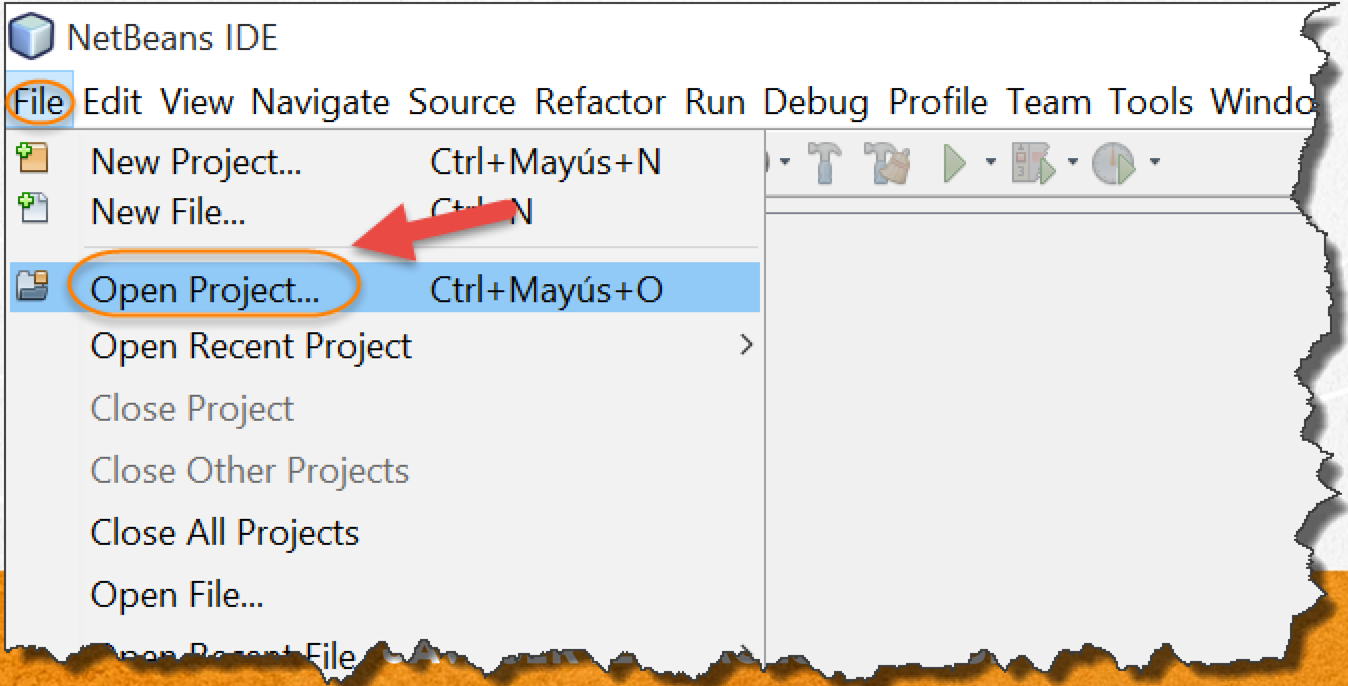
Name: John

Last Name: Smith

Desired Salary: 5000

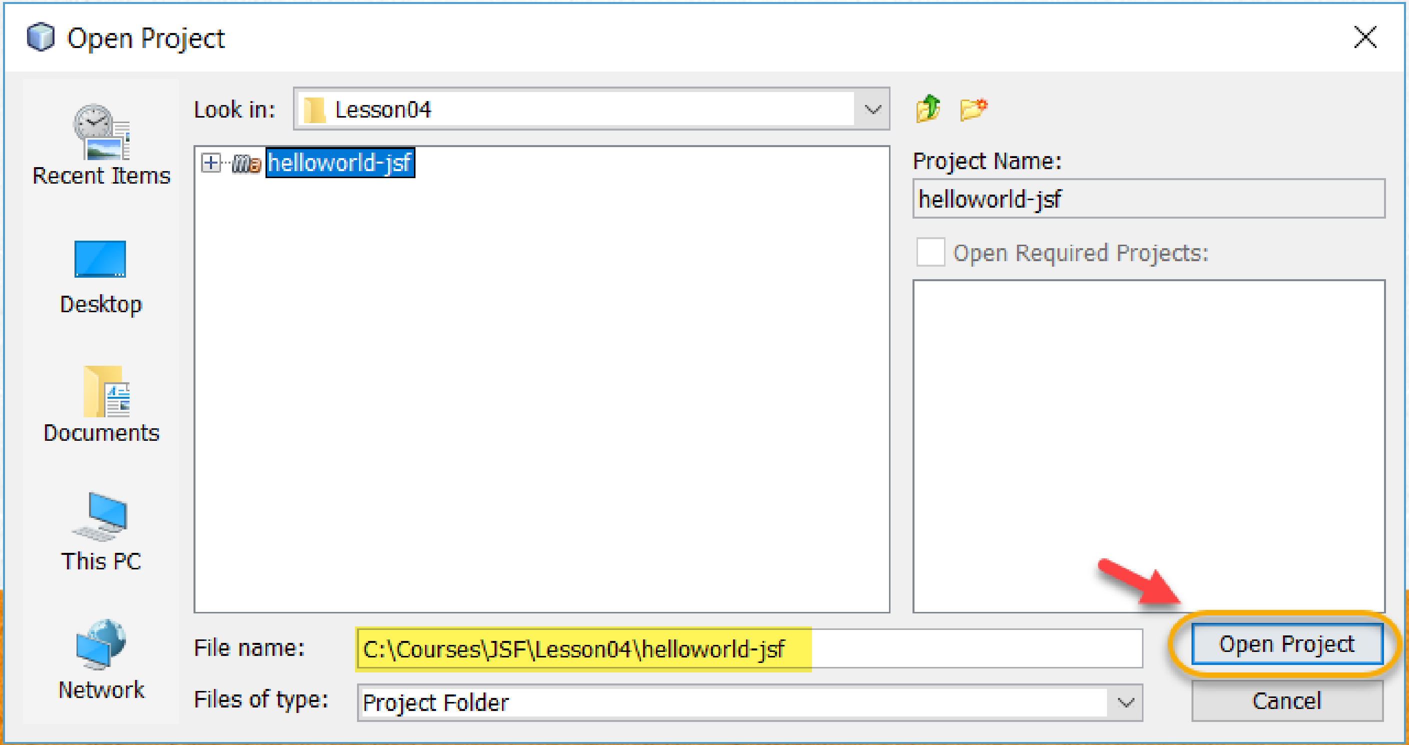
# 1. OPEN THE PROJECT

We open the project helloworld-jsf of the last lesson only in case we do not have it already open:


NetBeans IDE
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Windo
New Project... Ctrl+Mayús+N
New File... Ctrl+N
Open Project... Ctrl+Mayús+O
Open Recent Project
Close Project
Close Other Projects
Close All Projects
Open File...
Open Recent File

# 1. OPEN THE PROJECT

We open the project helloworld-jsf of the last lesson only in case we do not have it already open:



Look in: Lesson04

helloworld-jsf

Project Name: helloworld-jsf

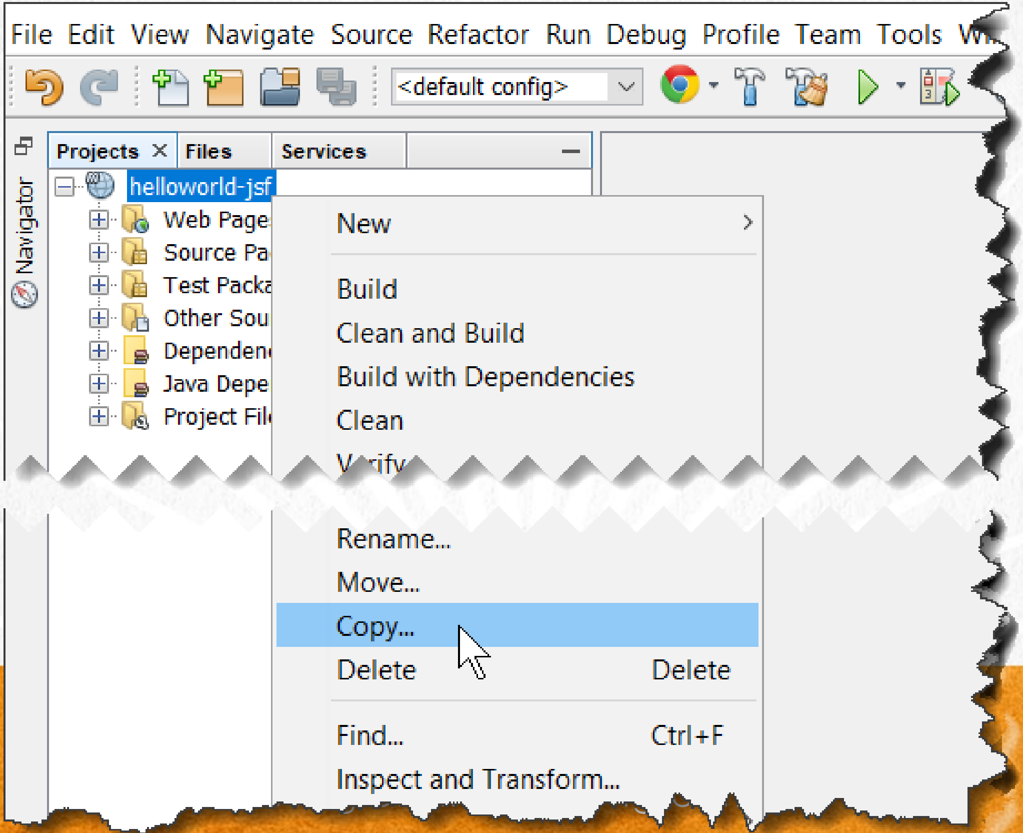
□ Open Required Projects:

File name: C:\Courses\JSF\Lesson04\helloworld-jsf

Files of type: Project Folder

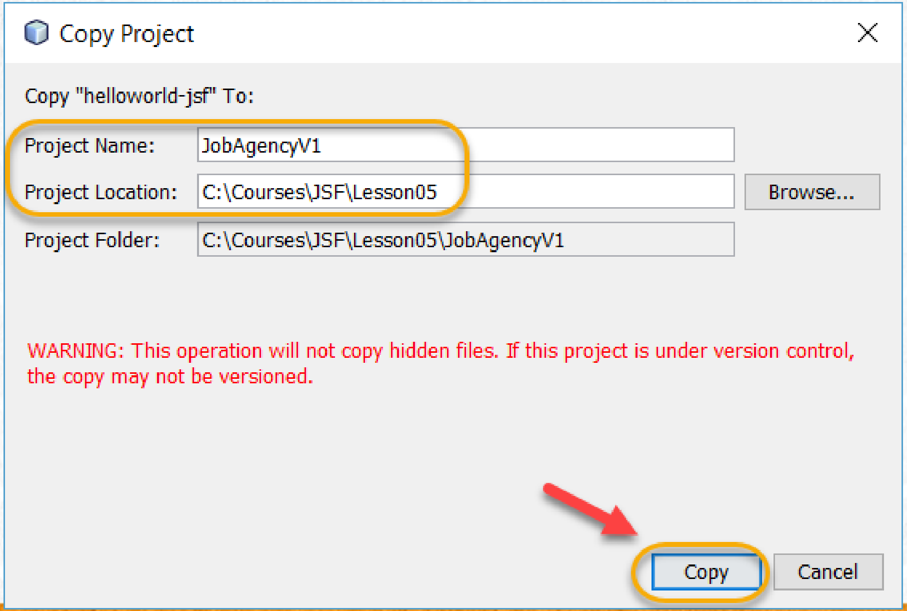
# 2. COPY THE PROJECT

We copy the project helloworld-jsf:



# 2. COPY THE PROJECT

We copy the project helloworld-jsf and rename it to JobAgencyV1:



Copy Project

Copy "helloworld-jsf" To:

Project Name: JobAgencyV1

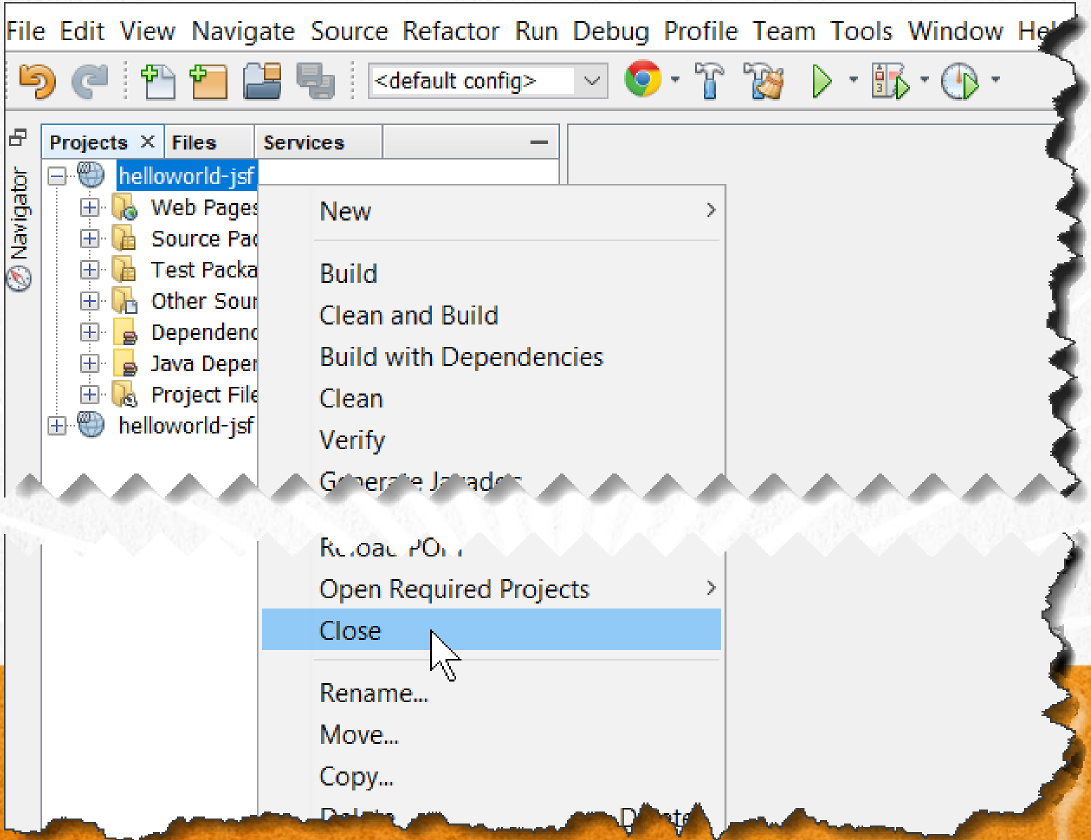
Project Location: C:\Courses\JSF\Lesson05

Project Folder: C:\Courses\JSF\Lesson05\JobAgencyV1

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

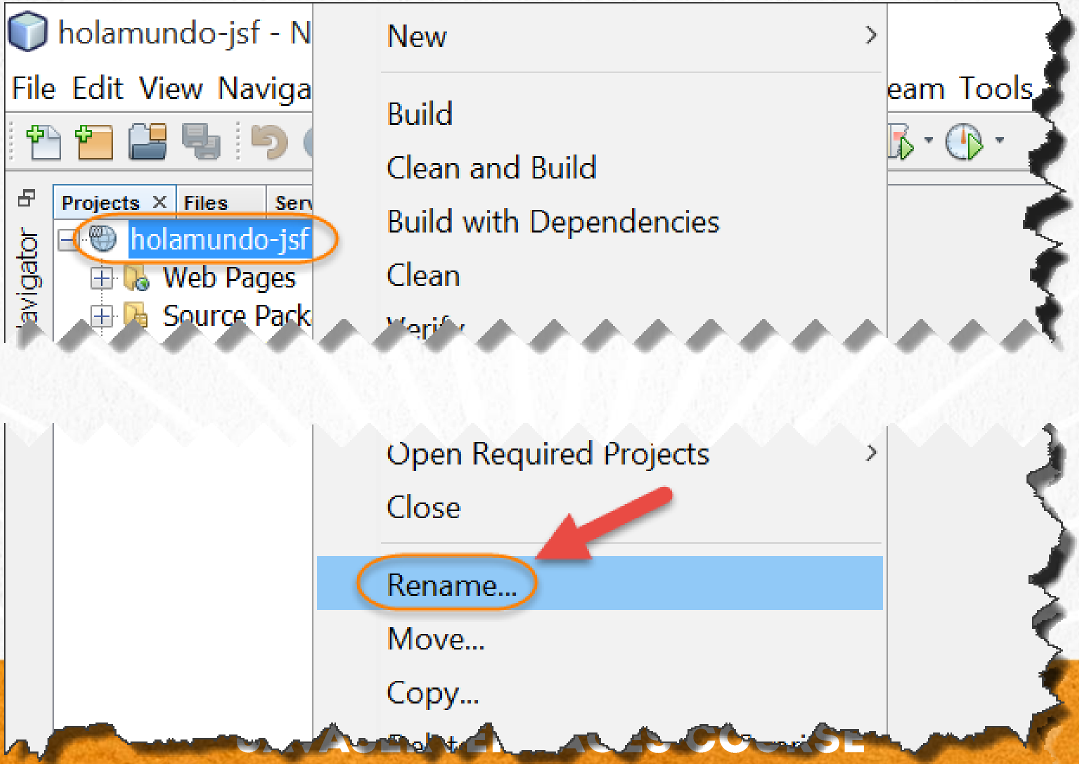
# 3. CLOSE THE PREVIOUS PROJECT

We closed the previous project, and left only the new open :



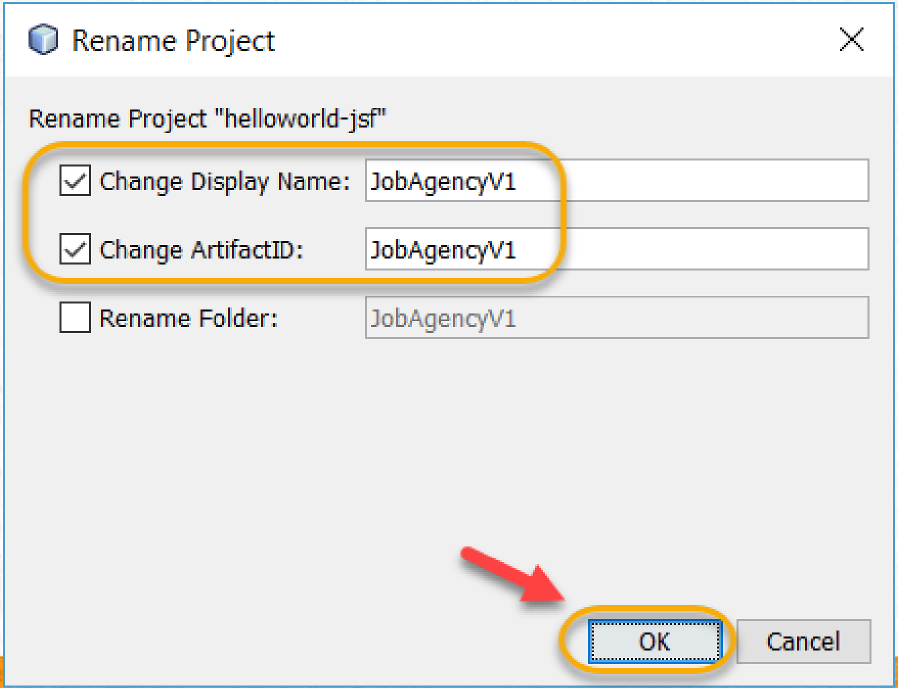
# 4. RENAME THE PROJECT

We change the name of the project to JobAgencyV1 :



# 4. RENAME THE PROJECT

We change the name of the project to JobAgencyV1:



Rename Project

Rename Project "helloworld-jsf"

☑ Change Display Name: JobAgencyV1

☑ Change ArtifactID: JobAgencyV1

□ Rename Folder: JobAgencyV1

# 5. MODIFY THE JAVA CLASS

We modified the Candidate.java file to add the last name field and the desired salary field, and create its getters / setters methods.

At the end the Candidate.java class will be as follows:

# 5. MODIFY THE CODE

## Candidate.java:

Click to download

```java
package beans.model;

import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import org.apache.logging.log4j.*;

@Named
@RequestScoped
public class Candidate {

    Logger log = LogManager.getRootLogger();

    private String name;
    private String lastName;
    private String desiredSalary;

    public Candidate() {
        log.info("Creating the Candidate object");
        this.setName("Introduce your name");
    }

    public String getName() {
        return name;
    }
```

# 5. MODIFY THE CODE

Candidate.java:

Click to download

```java
    public void setName(String name) {
        this.name = name;
        log.info("Modifying the name property:" + this.name);
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
        log.info("Modifying the lastName property:" + this.name);
    }

    public String getDesiredSalary() {
        return desiredSalary;
    }

    public void setDesiredSalary(String desiredSalary) {
        this.desiredSalary = desiredSalary;
        log.info("Modifying the desiredSalary property:" + this.desiredSalary);
    }
}
```

# 6. MODIFY THE XHTML FILE

## Modify the index.xhtml file to validate the name field:

In the index.xhtml file, replace :

```
<h:inputText id="name" value="#{candidate.name}" />
```

- By the following code :

```
<h:inputText id="name" required="true" value="#{candidate.name}" />
```

- In this way we will validate that the name field contains information before sending the form.

# 6. MODIFY THE XHTML FILE

## Modify the index.xhtml file to add the lastName field :

- Add the following code in index.xhtml generating a new line <tr>:

```
<tr>
   <td><h:outputLabel for="lastName" value="Last Name:"  /></td>
   <td><h:inputText id="lastName" required="true"
                    value="#{candidate.lastName}" /></td>
   <td><h:message for="lastName" /></td>
 </tr>
```

# 6. MODIFY THE XHTML FILE

## Modify the index.xhtml file to add the desiredSalary field :

- Add the following code in index.xhtml generating a new line <tr>:

```
<tr>
   <td><h:outputLabel for="desiredSalary" value="Desired Salary:"  /></td>
   <td><h:inputText id="desiredSalary" required="true"
                    value="#{candidate.desiredSalary}">
       <f:validateLongRange minimum="500" maximum="5000"/>
     </h:inputText></td>
   <td><h:message for="desiredSalary" /></td>
 </tr>
```

# 6. MODIFY THE XHTML FILE

Finally, by making the previous changes and some other minor changes, such as changing the title of the page and adding the taglib of core f to the beginning of the index.xhtml file.

```
xmlns:f="http://xmlns.jcp.org/jsf/core"
```

And we add the global message handling component in the html form:

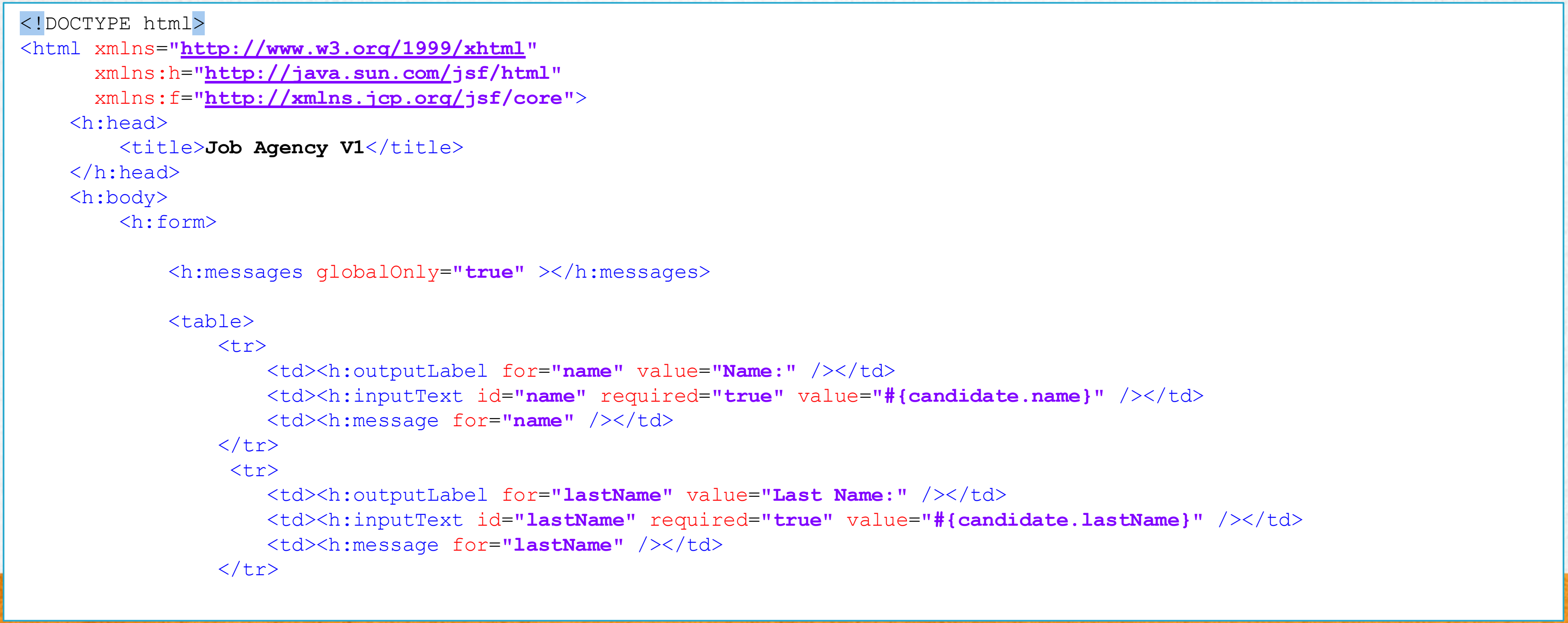
```
<h:messages globalOnly="true" ></h:messages>
```

The final index.xhtml file is as follows:

# 7. MODIFY THE FILE

index.xhtml:

Click to download



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Job Agency V1</title>
    </h:head>
    <h:body>
        <h:form>

            <h:messages globalOnly="true" ></h:messages>

            <table>
                <tr>
                    <td><h:outputLabel for="name" value="Name:" /></td>
                    <td><h:inputText id="name" required="true" value="#{candidate.name}" /></td>
                    <td><h:message for="name" /></td>
                </tr>
                 <tr>
                    <td><h:outputLabel for="lastName" value="Last Name:" /></td>
                    <td><h:inputText id="lastName" required="true" value="#{candidate.lastName}" /></td>
                    <td><h:message for="lastName" /></td>
                </tr>
```

# 7. MODIFY THE FILE

## index.xhtml:

Click to download

```
               <tr>
                    <td><h:outputLabel for="desiredSalary" value="Desired Salary:"  /></td>
                    <td><h:inputText id="desiredSalary" required="true"
                                     value="#{candidate.desiredSalary}">
                            <f:validateLongRange minimum="500" maximum="5000"/>
                        </h:inputText></td>
                    <td><h:message for="desiredSalary" /></td>
                </tr>
            </table>
            <h:commandButton action="#{vacantForm.send}" value="Send" />
        </h:form>
    </h:body>
</html>
```

# 8. MODIFY THE JAVA CLASS

Modify the VacanteForm.java class to add validation but now with Java code.

In case that the specified condition is met, we send a message to the user.

This is just an example, but we can add any presentation logic to our ManagedBean so we can add the necessary validations to our presentation layer with JSF.

Let's see how the class VacanteForm.java is finally

# 8. MODIFY THE CODE

## VacantForm.java:

Click to download

```java
package beans.backing;

import beans.model.Candidate;
import javax.inject.*;
import javax.enterprise.context.RequestScoped;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import org.apache.logging.log4j.*;

@Named
@RequestScoped
public class VacantForm {

    Logger log = LogManager.getRootLogger();

    @Inject
    private Candidate candidate;

    public VacantForm() {
        log.info("Creating VacantForm object");
    }

    public void setCandidate(Candidate candidate) {
        this.candidate = candidate;
    }
```

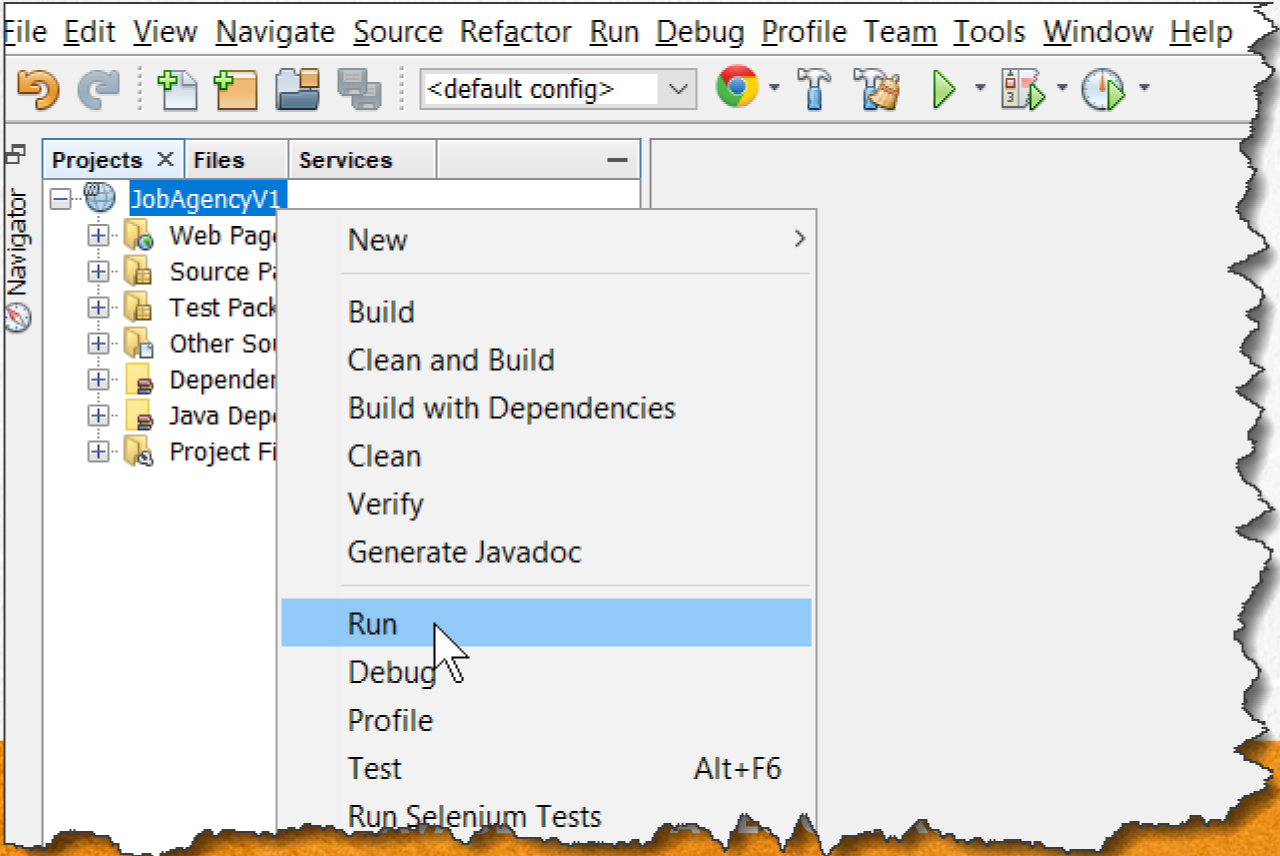
# 8. MODIFY THE CODE

## VacantForm.java:

Click to download

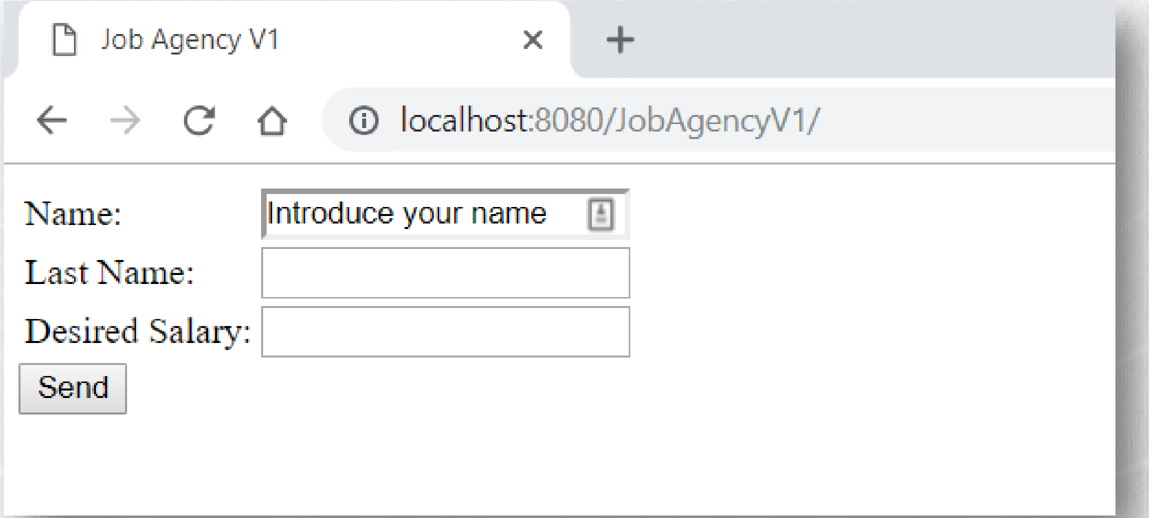
```
    public String send() {
        log.info("send() Name=" + this.candidate.getName());
        log.info("send() Desired Salary=" + this.candidate.getDesiredSalary());

        if (this.candidate.getName().equals("John")) {

            if (this.candidate.getLastName().equals("Smith")) {
                String msg = "Thanks, but John Smith is already working with us.";
                FacesMessage facesMessage = new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, msg);
                FacesContext facesContext = FacesContext.getCurrentInstance();
                String componentId = null; //This is a global message
                facesContext.addMessage(componentId, facesMessage);
                return "index";
            }
            return "success";//success.xhtml
        } else {
            return "failure"; //failure.xhtml
        }
    }
}
```

# 8. EXECUTE THE PROJECT



# 8. EXECUTE THE PROJECT

The output of the project is as follows:



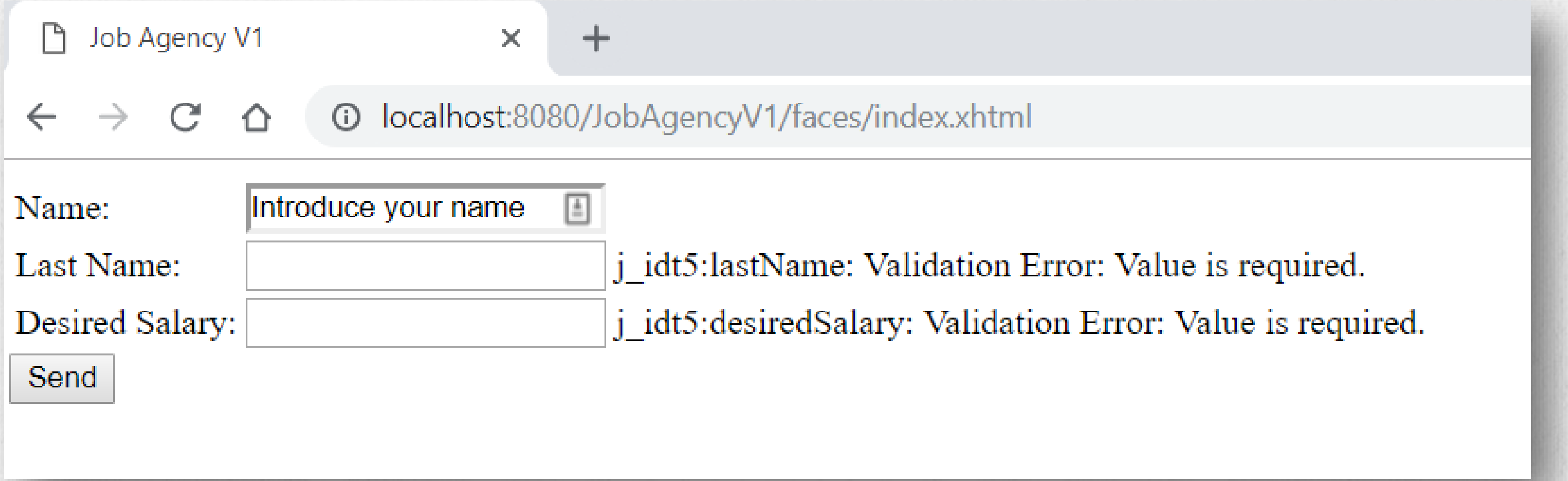
Name: Introduce your name

Last Name:

Desired Salary:

# 8. EXECUTE THE PROJECT

We test with different values:



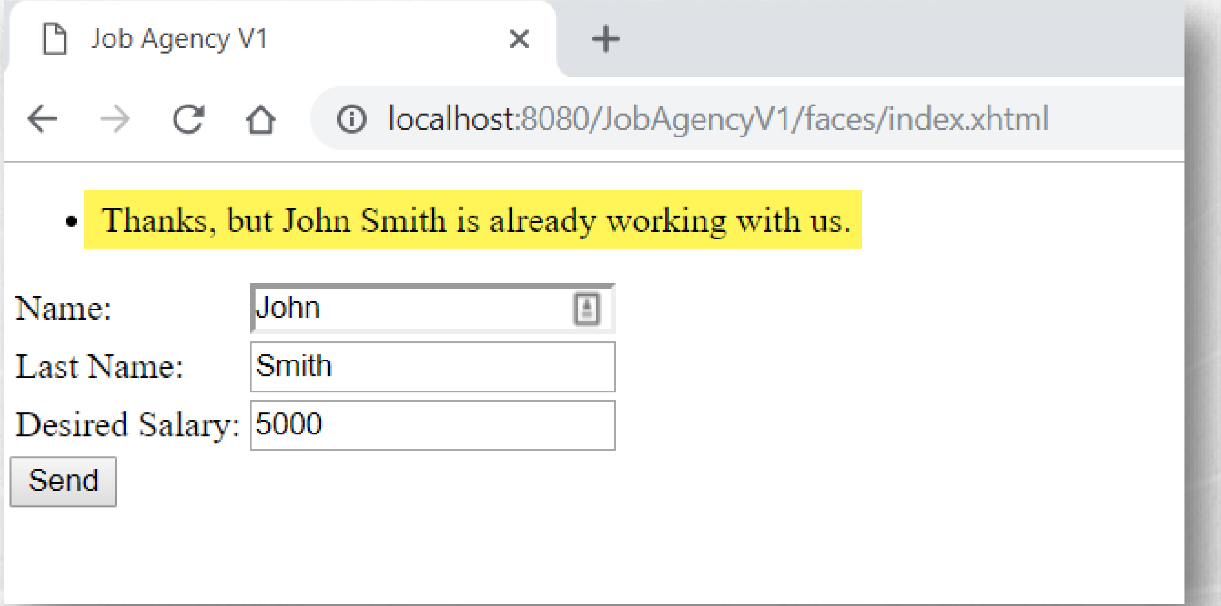
Name: Introduce your name

Last Name: j_idt5:lastName: Validation Error: Value is required.

Desired Salary: j_idt5:desiredSalary: Validation Error: Value is required.

# 8. EXECUTE THE PROJECT

Validation case in ManagedBean:



- Thanks, but John Smith is already working with us.

|  |  |
|---|---|
| Name: | John |
| Last Name: | Smith |
| Desired Salary: | 5000 |

# IN CASE OF PROBLEMS

1. Stop Glassfish server
2. Execute Clean & Build again
3. Run the application

- Repeat the steps above if apply any changes to the code and find errors in the application.

- If the problem is not solved, you can try loading the resolved project, which is 100% functional

# EXTRA TASKS

It is left as extra tasks to test more cases with the form and verify that the validations are applied correctly.

You can also do a step-by-step review by applying the debug mode.

Finally, you should check the server log to see that everything works as expected and also see when the validations are applied and in which phase of the life cycle of JSF.

# EXERCISE CONCLUSION

In this lesson we studied the validators topic in JSF.

We saw several ways to validate the fields of our JSF form, besides we studied how to add validation from a ManagedBean, and in this way not only validate the form, but also with presentation logic within our Java code with JSF.

With this we already have the bases to handle the issue of validators in JSF.

ONLINE COURSE

# JAVASERVER FACES (JSF)

By: Ing. Ubaldo Acosta

Global Mentoring

JAVASERVER FACES COURSE

www.globalmentoring.com.mx