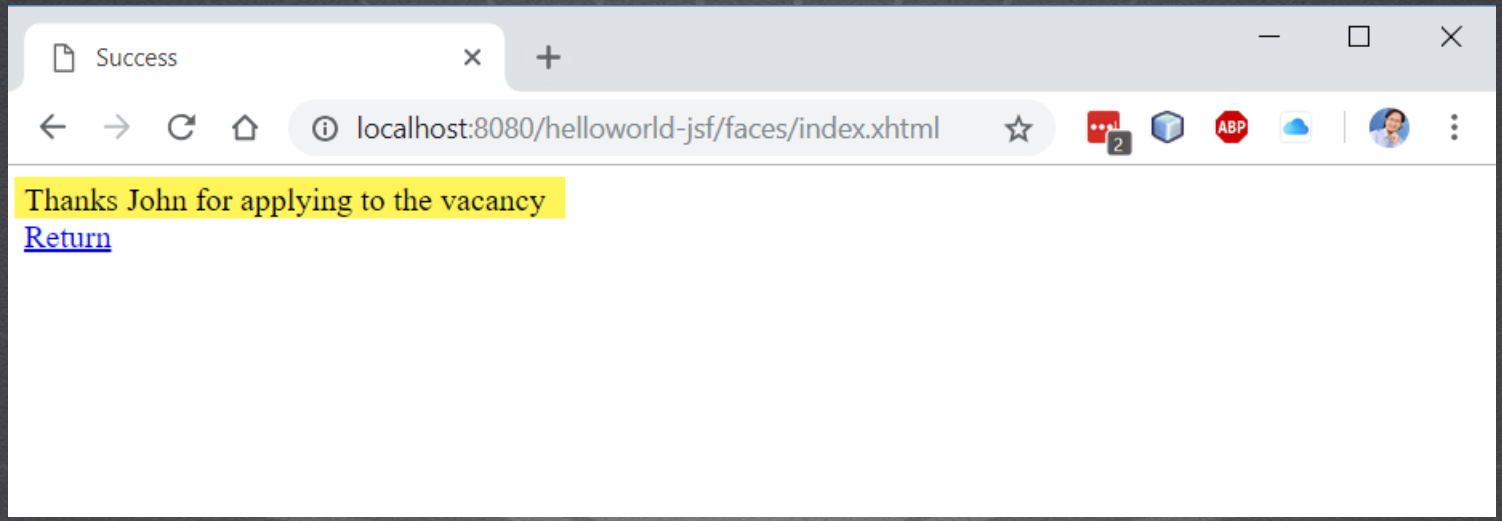
# EXERCISE

# BACKING MANAGED BEAN IN JSF

# EXERCISE OBJECTIVE

Modify the project of helloworld-jsf to apply the concept of Backing ManagedBean. The final result is as follows:



# EXERCISE OBJECTIVE

We will introduce the concept of Backing Managed Bean to our application.

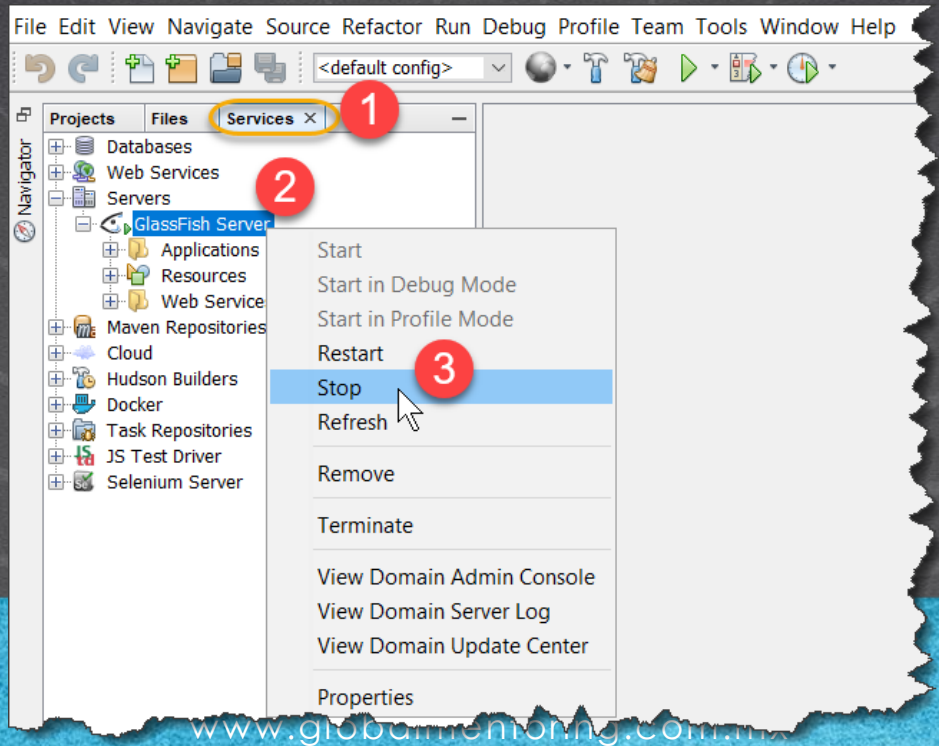
We will use the concept of annotations to inject dependencies between Managed Bean's.

Additionally we will add some JSF pages with the respective navigation rules.

Finally we will associate functionality to the Send button to execute the application flow.

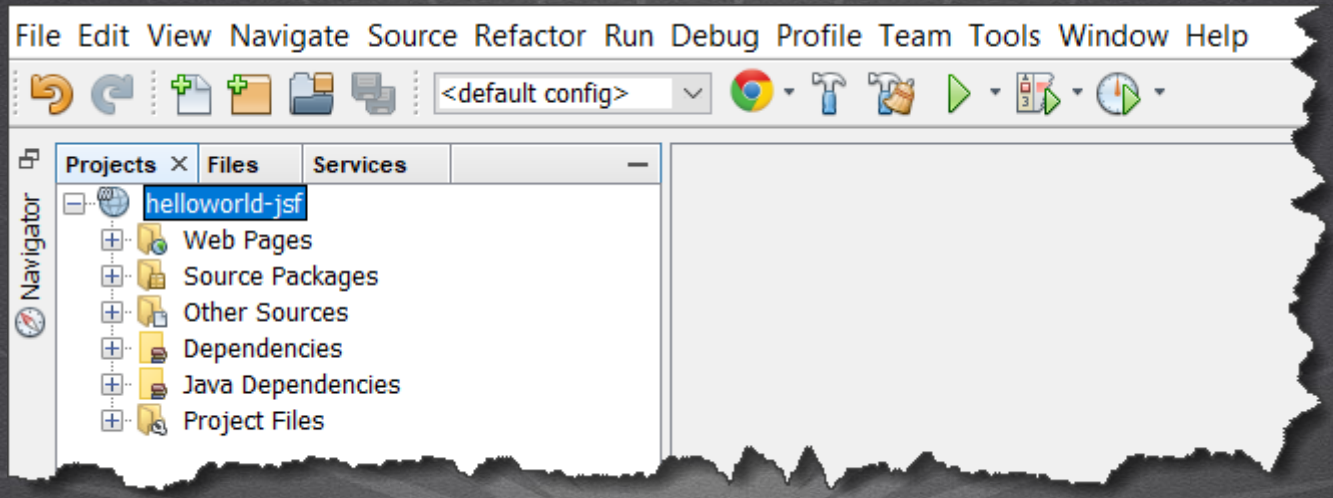
# BEFORE STARTING

We continue working with the project of the previous lesson. If the Glassfish server is up, we will stop it, otherwise, with every change we make, the application will be deployed again.



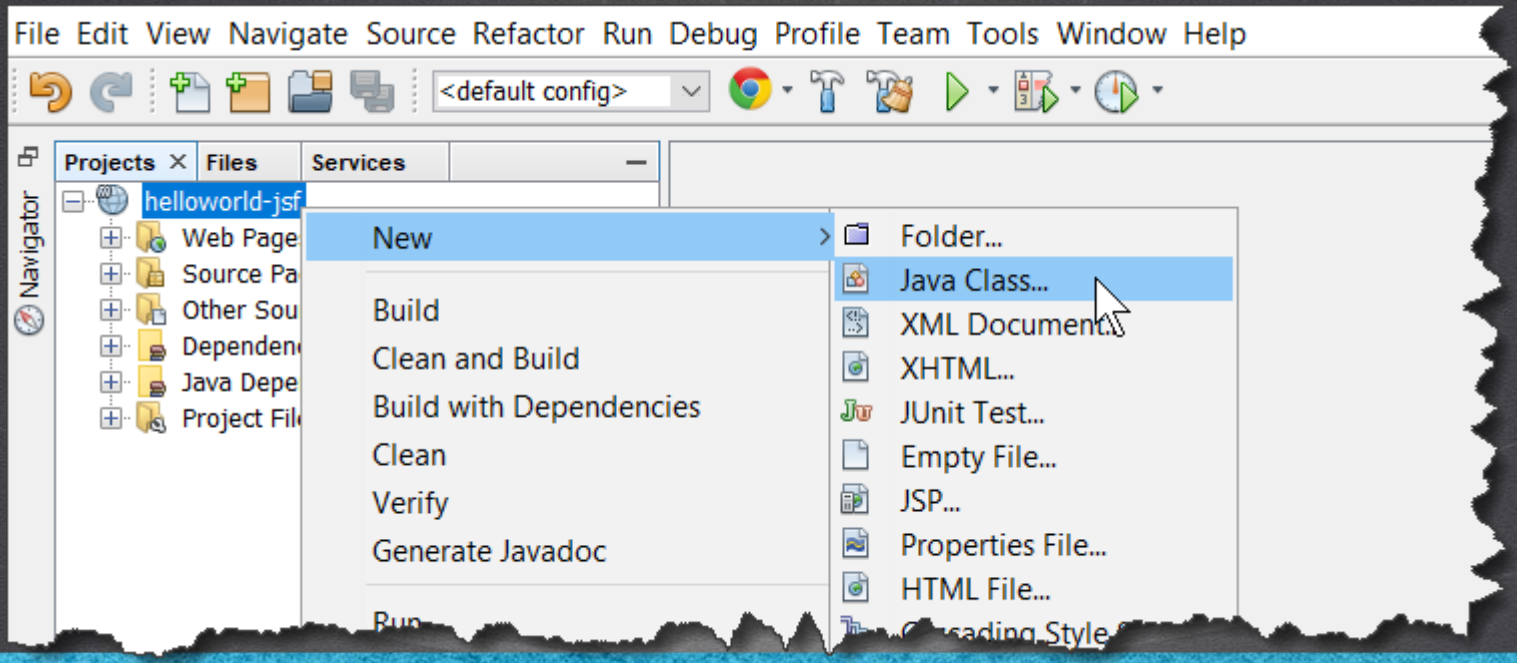
# 1. OPEN THE PROJECT

We continue working with the project of the previous lesson, so it must already be open :



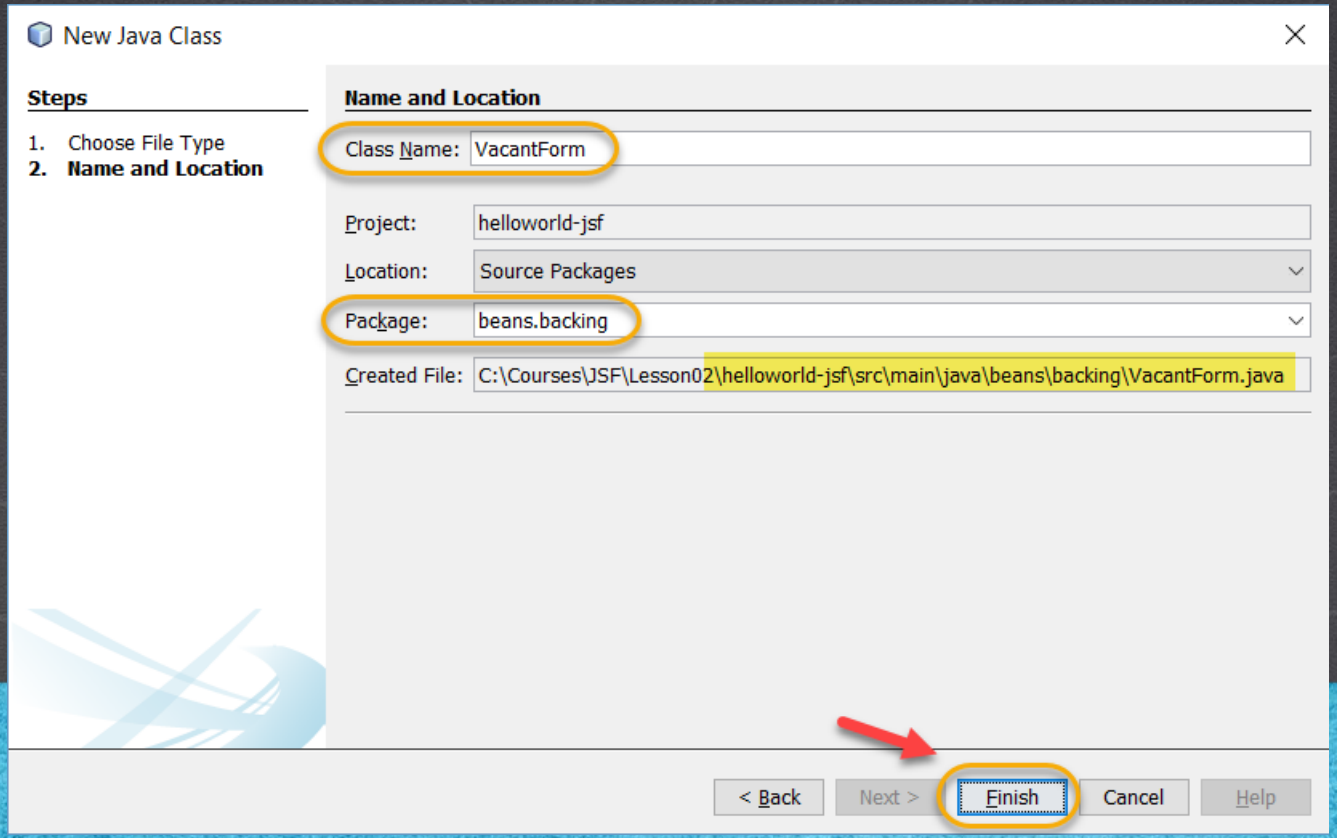
# 2. CREATE A JAVA CLASS

We create a Java class that will be a ManagedBean of JSF, called VacantForm.java :



# 2. CREATE A JAVA CLASS

We create a Java class that will be a ManagedBean of JSF, called VacantForm.java :



# 3. MODIFY THE CODE

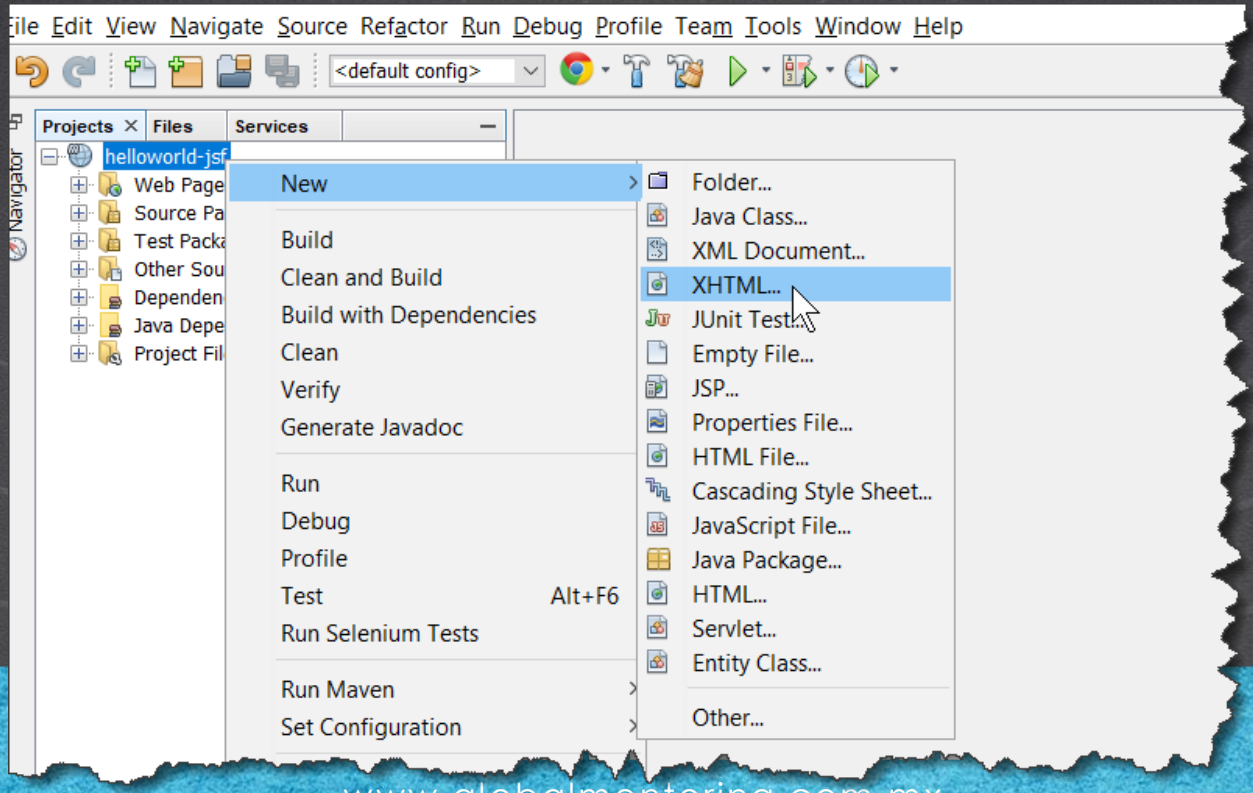
## VacantForm.java:

Click to download

```
package beans.backing;

import beans.model.Candidate;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;

@Named
@RequestScoped
public class VacantForm {

  @Inject
  private Candidate candidate;

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

  public String send() {
    if (this.candidate.getName().equals("John")) {
      return "success";
    } else {
      return "failure";
    }
  }
}
```

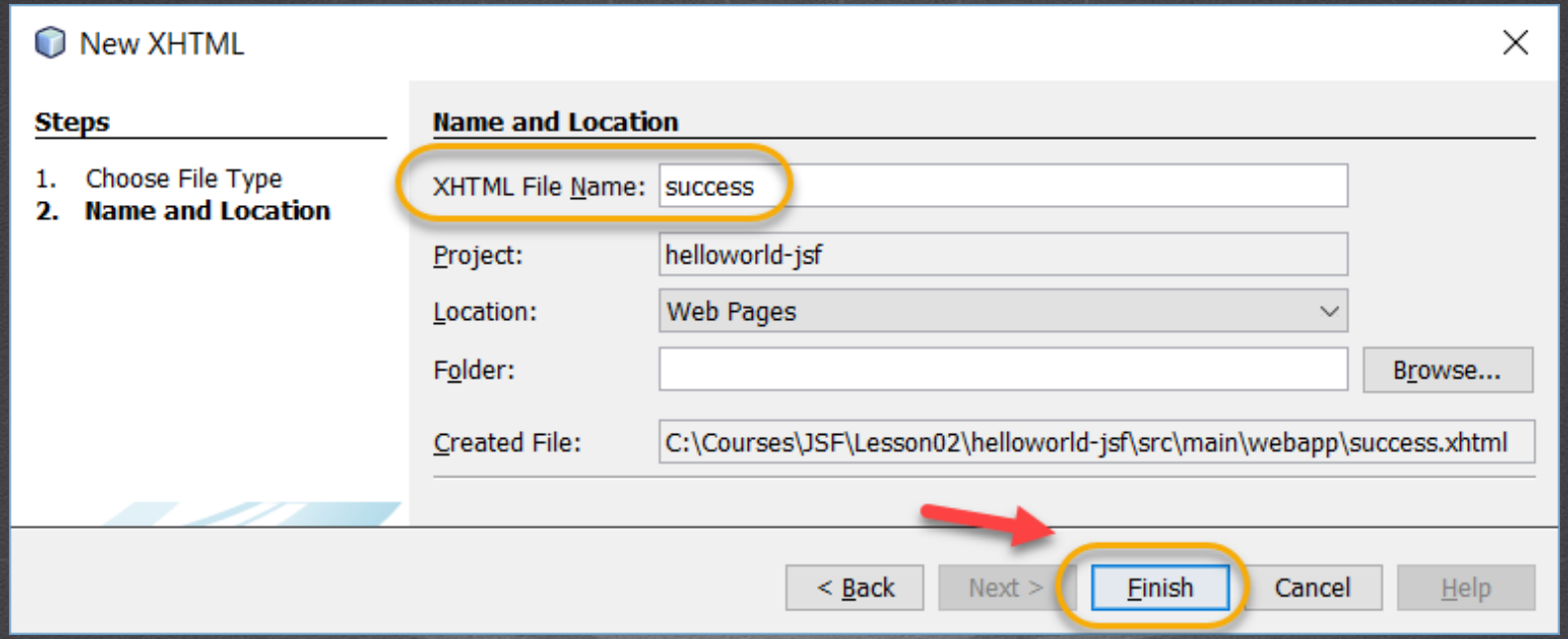
# 4. ADD A JSF PAGE FOR THE CASE OF "SUCCESS"

We created an .xhtml page for the case of success.



# 4. ADD A JSF PAGE FOR THE CASE OF "SUCCESS

We created an .xhtml page for the case of success.



# 5. MODIFY THE CODE

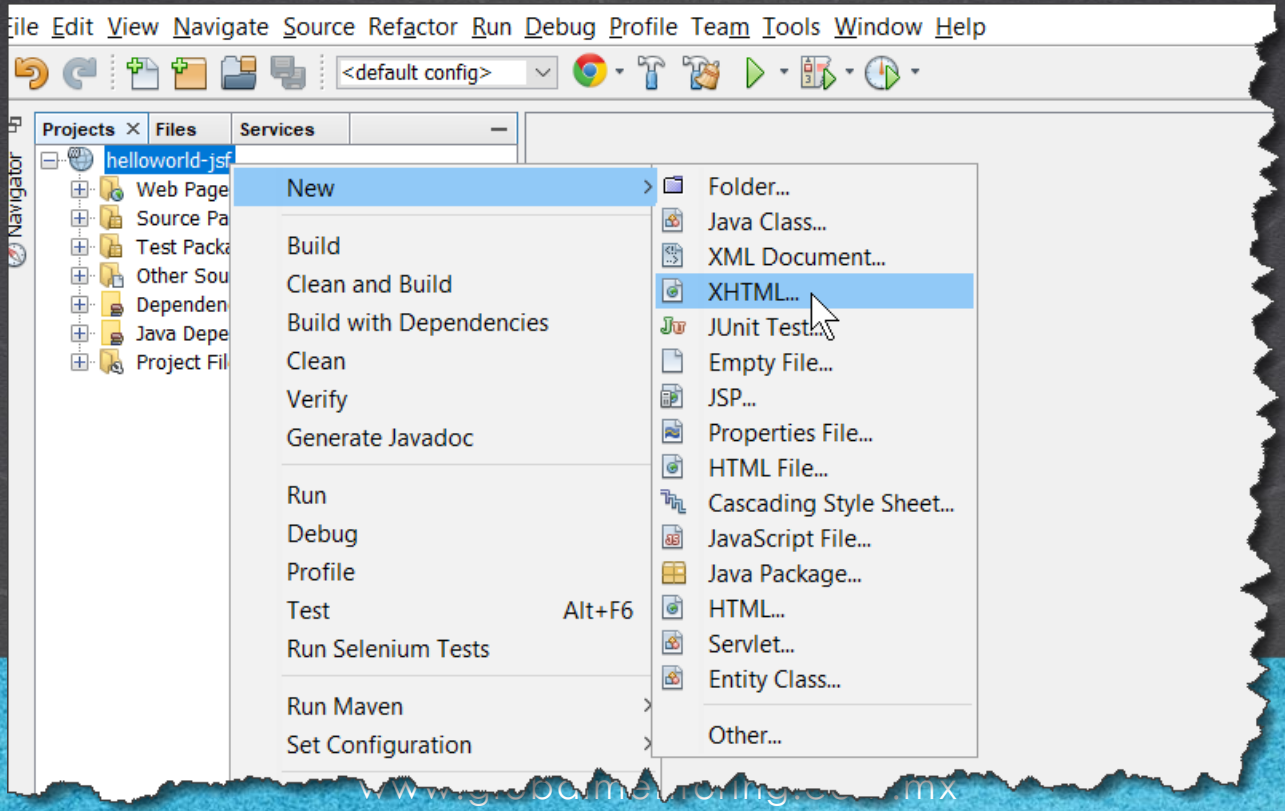
success.xhtml:

Click to download

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Success</title>
  </h:head>
  <h:body>
    <h:form>
      Thanks #{candidate.name} for applying to the vacancy
      <br/>
      <h:commandLink action="index">Return</h:commandLink>
    </h:form>
  </h:body>
</html>
```

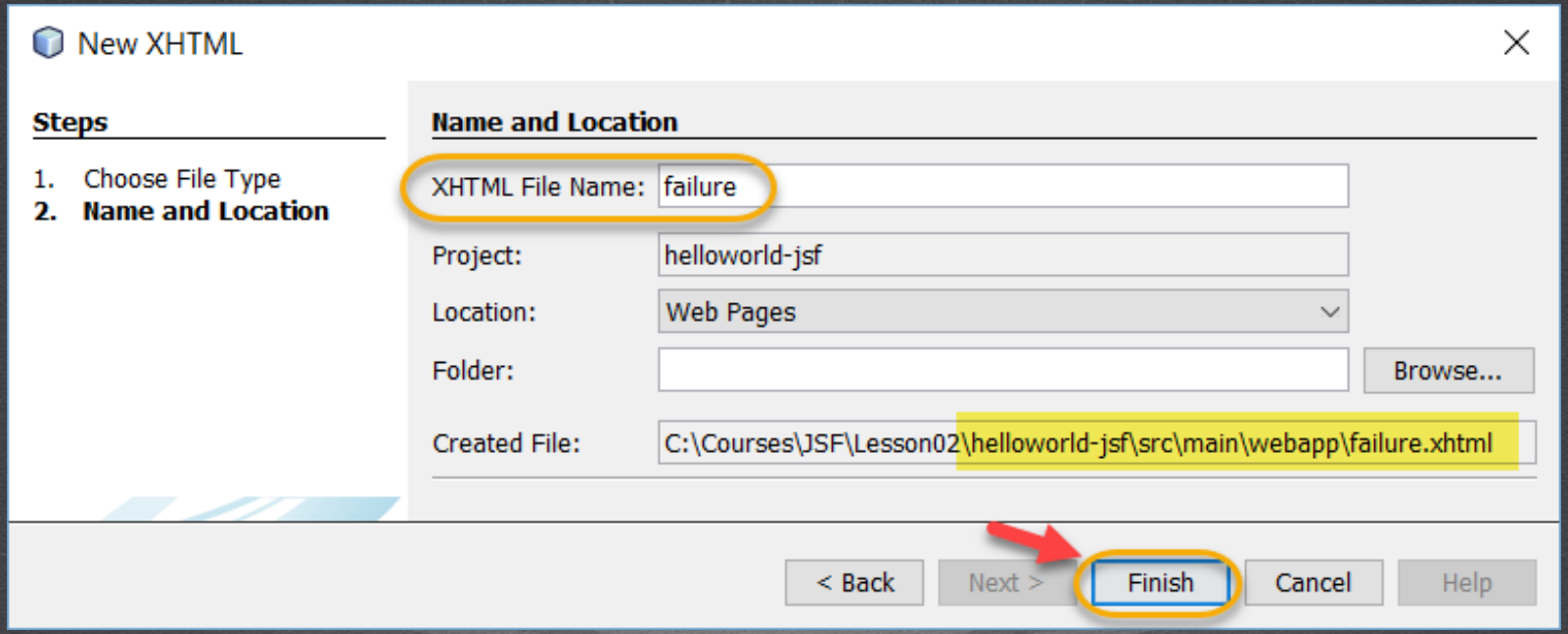
# 6. ADD A JSF PAGE FOR THE CASE OF "FAILURE"

We created an .xhtml page for the case of failure.



# 6. ADD A JSF PAGE FOR THE CASE OF "FAILURE"

We created an .xhtml page for the case of failure.



# 7. MODIFY THE CODE

failure.xhtml:

Click to download

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Failure</title>
  </h:head>
  <h:body>
    <h:form>
      Sorry #{candidate.name}, all vacancies have already been filled.
      <br/>
      <h:commandLink action="index">Return</h:commandLink>
    </h:form>
  </h:body>
</html>
```

# 8. MODIFY THE INDEX.XHTML PAGE

- **Locate in the content of the index.xhtml page, the following:**

```
<h:commandButton value="Send" />
```

- **Replace it with the following code:**

```
<h:commandButton action="#{vacantForm.send}" value="Send" />
```

# 8. MODIFY THE CODE

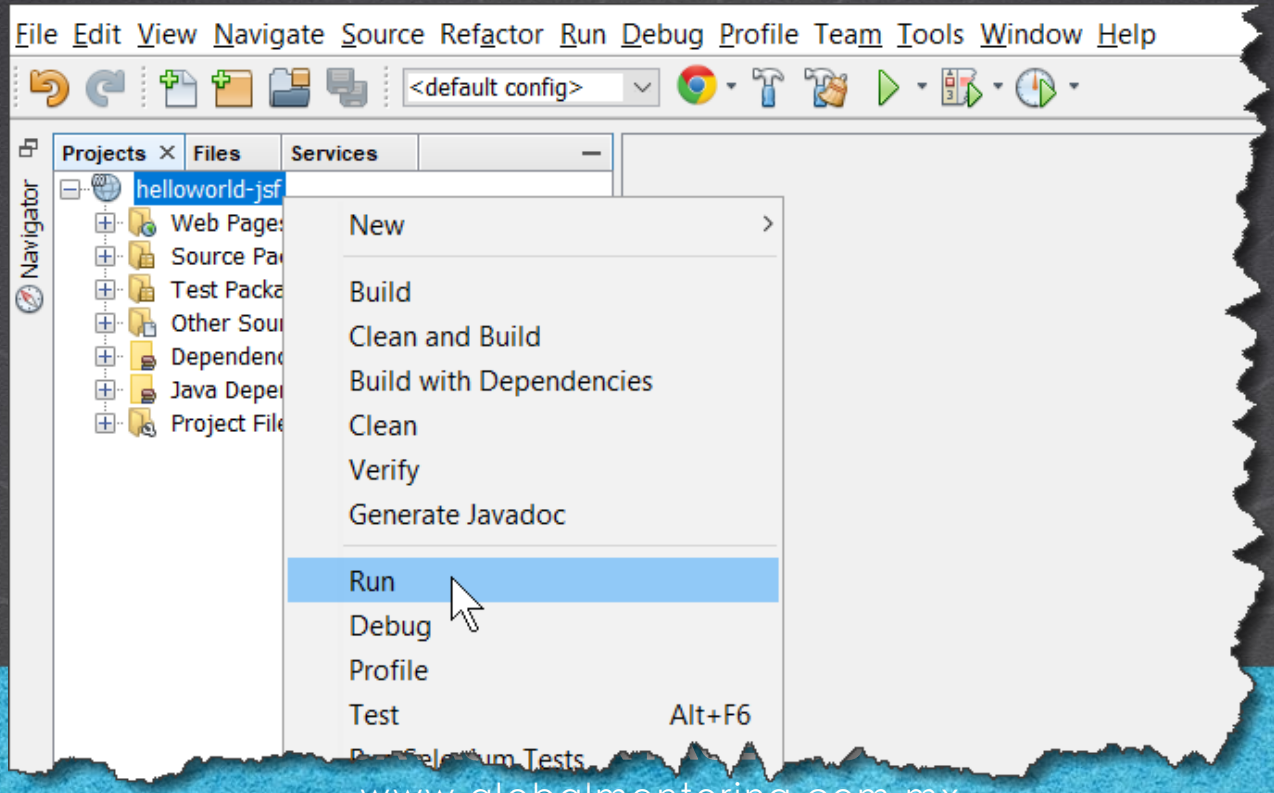
index.xhtml:

Click to download

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>HelloWorld JSF</title>
    </h:head>
    <h:body>
        <h:form>
            <table>
                <tr>
                    <td><h:outputLabel for="name" value="Name:" /></td>
                    <td><h:inputText id="name" value="#{candidate.name}" /></td>
                    <td><h:message for="name" /></td>
                </tr>
            </table>
            <h:commandButton action="#{vacantForm.send}" value="Send" />
        </h:form>
    </h:body>
</html>
```

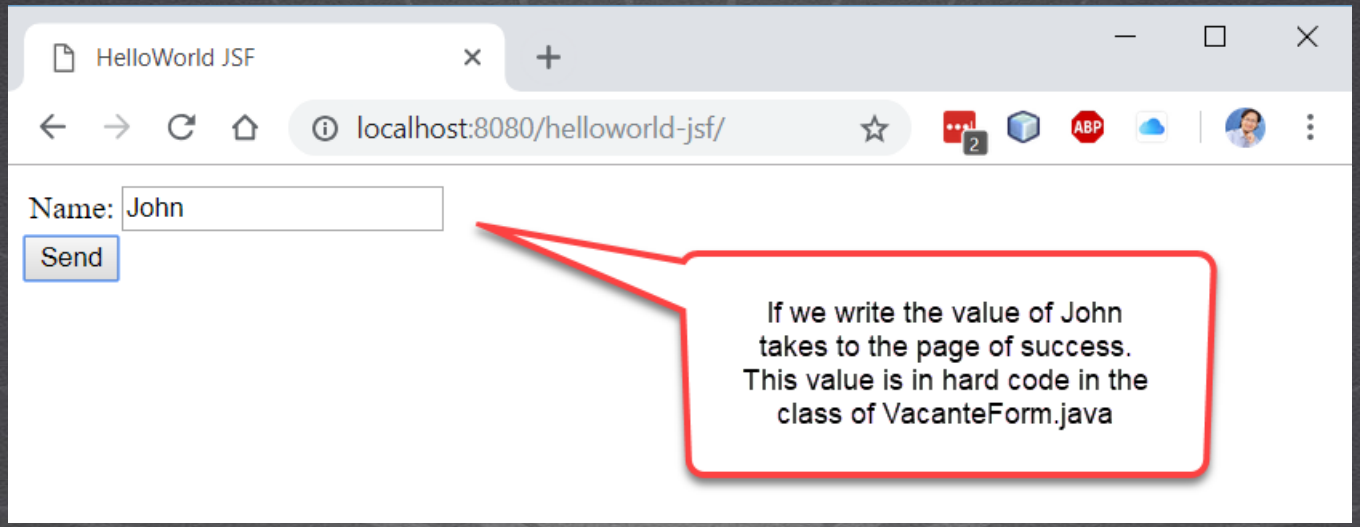
# 9. EXECUTE THE PROJECT

Execute the Project:



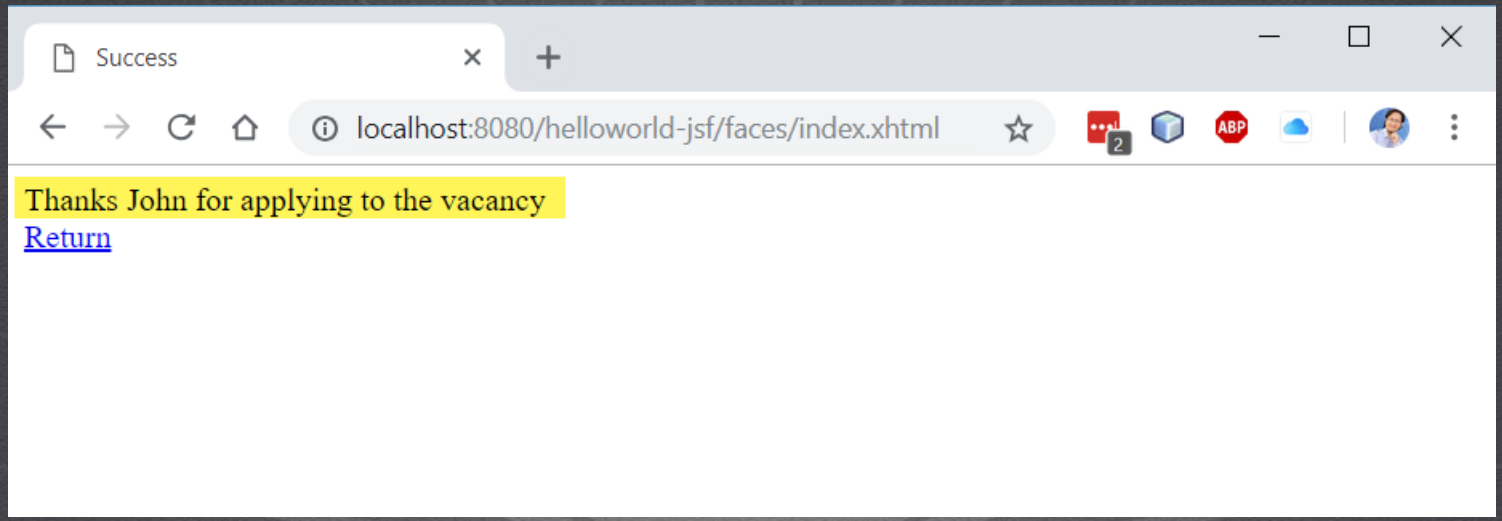
# 9. EXECUTE THE PROJECT

We execute the project. We test the success case, providing the value of Juan:



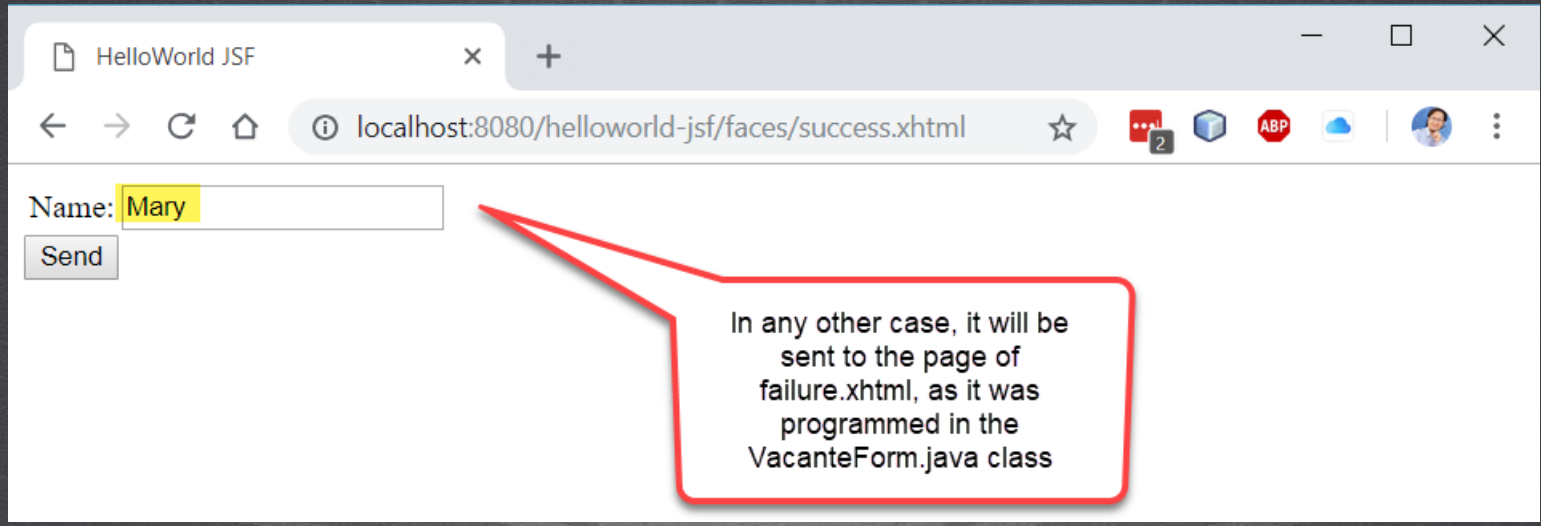
# 9. EXECUTE THE PROJECT

We execute the project. We test the success case, providing the value of Juan:



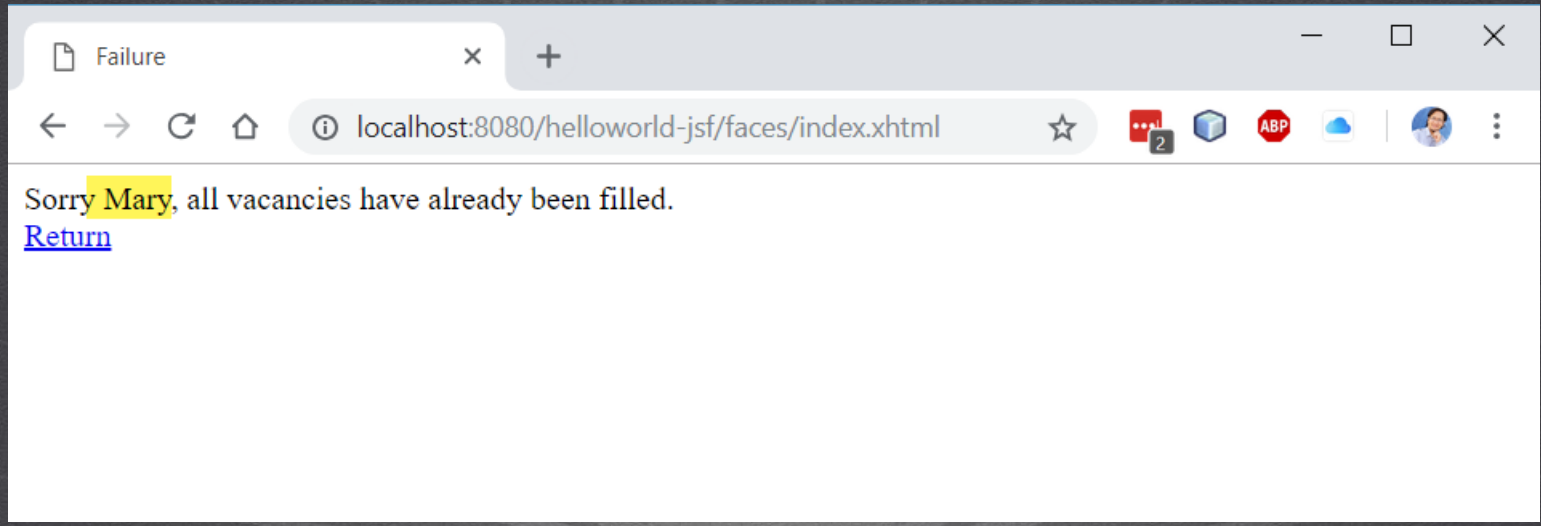
# 9. EXECUTE THE PROJECT

We execute the project. We test the failure case by providing a different name to John.



# 9. EXECUTE THE PROJECT

We execute the project. We test the failure case by providing a different name to John.



# IN CASE OF PROBLEMS

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

- Reapeat 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

# EXERCISE CONCLUSION

- With this exercise we have added the JSF Backing ManagedBean concept. Recall that this concept is a JSF Bean that will have navigation logic and some other logic of the application with respect to the presentation layer. Unlike ManagedBean, we will not use it for this purpose, but only to associate properties of a bean class.

- We also added two JSF pages to execute two cases, one of “success” and the other of “failure”. If the given name is equal to "John", then the success page is displayed, otherwise the failure page is displayed.

- In this case, we are applying the default navigation of JSF, without the need to add the navigation cases to the faces-config.xml file, since the name of the page is equal to the name that returns the navigation flow in the VacantForm.send() method.

ONLINE COURSE

# JAVASERVER FACES (JSF)

By: Eng. Ubaldo Acosta