EXERCISE

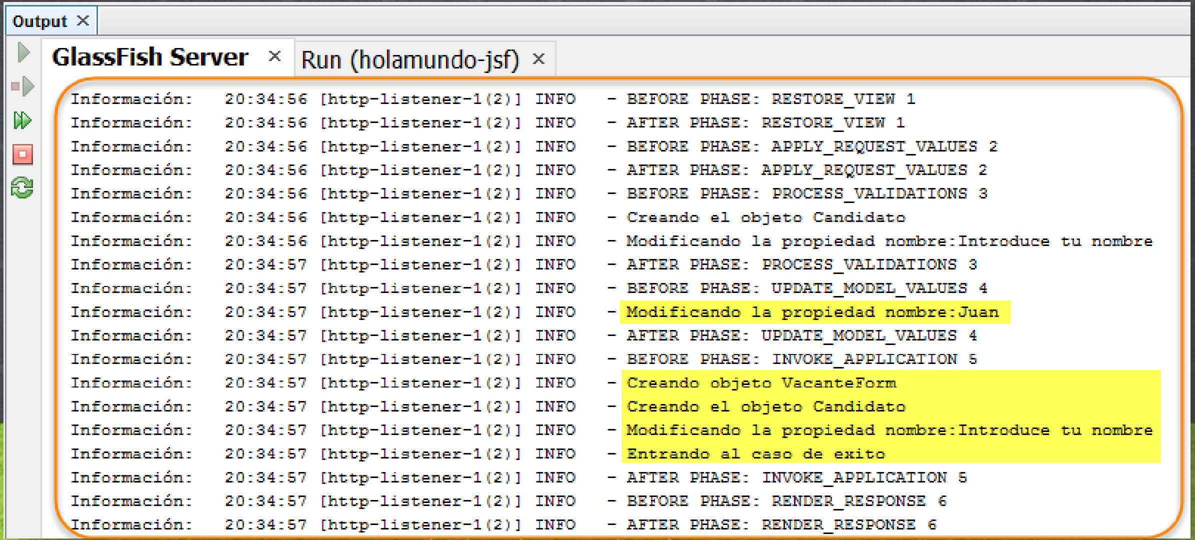
LIFE CYCLE IN JSF

Global Mentoring

EXERCISE OBJECTIVE

In this exercise we will create the necessary classes to show the life cycle in the console of our Java EE server. We will use log4j to show in the console the result of the life cycle and each of its stages.

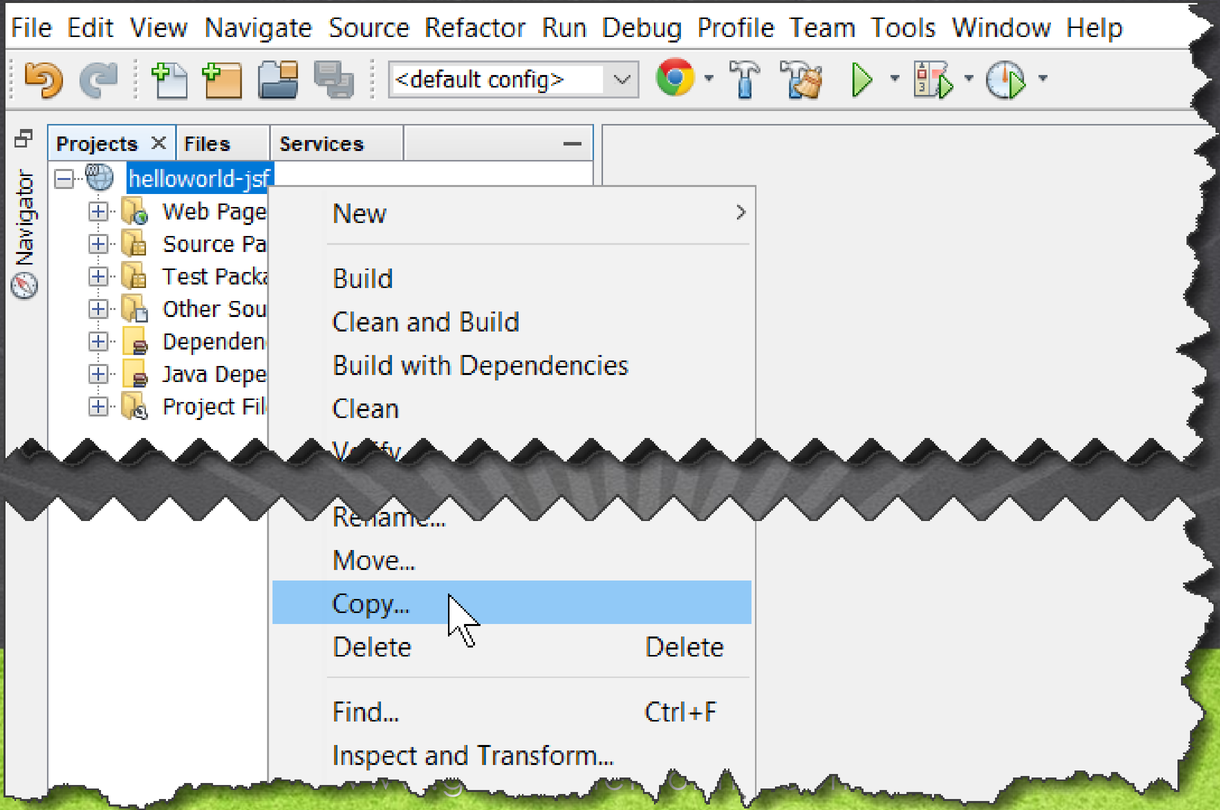
The result should be similar to the one shown below:



```
Información:   20:34:56 [http-listener-1(2)] INFO   - BEFORE PHASE: RESTORE_VIEW 1
Información:   20:34:56 [http-listener-1(2)] INFO   - AFTER PHASE: RESTORE_VIEW 1
Información:   20:34:56 [http-listener-1(2)] INFO   - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Información:   20:34:56 [http-listener-1(2)] INFO   - AFTER PHASE: APPLY_REQUEST_VALUES 2
Información:   20:34:56 [http-listener-1(2)] INFO   - BEFORE PHASE: PROCESS_VALIDATIONS 3
Información:   20:34:56 [http-listener-1(2)] INFO   - Creando el objeto Candidato
Información:   20:34:56 [http-listener-1(2)] INFO   - Modificando la propiedad nombre:Introduce tu nombre
Información:   20:34:57 [http-listener-1(2)] INFO   - AFTER PHASE: PROCESS_VALIDATIONS 3
Información:   20:34:57 [http-listener-1(2)] INFO   - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Información:   20:34:57 [http-listener-1(2)] INFO   - Modificando la propiedad nombre:Juan
Información:   20:34:57 [http-listener-1(2)] INFO   - AFTER PHASE: UPDATE_MODEL_VALUES 4
Información:   20:34:57 [http-listener-1(2)] INFO   - BEFORE PHASE: INVOKE_APPLICATION 5
Información:   20:34:57 [http-listener-1(2)] INFO   - Creando objeto VacanteForm
Información:   20:34:57 [http-listener-1(2)] INFO   - Creando el objeto Candidato
Información:   20:34:57 [http-listener-1(2)] INFO   - Modificando la propiedad nombre:Introduce tu nombre
Información:   20:34:57 [http-listener-1(2)] INFO   - Entrando al caso de exito
Información:   20:34:57 [http-listener-1(2)] INFO   - AFTER PHASE: INVOKE_APPLICATION 5
Información:   20:34:57 [http-listener-1(2)] INFO   - BEFORE PHASE: RENDER_RESPONSE 6
Información:   20:34:57 [http-listener-1(2)] INFO   - AFTER PHASE: RENDER_RESPONSE 6
```

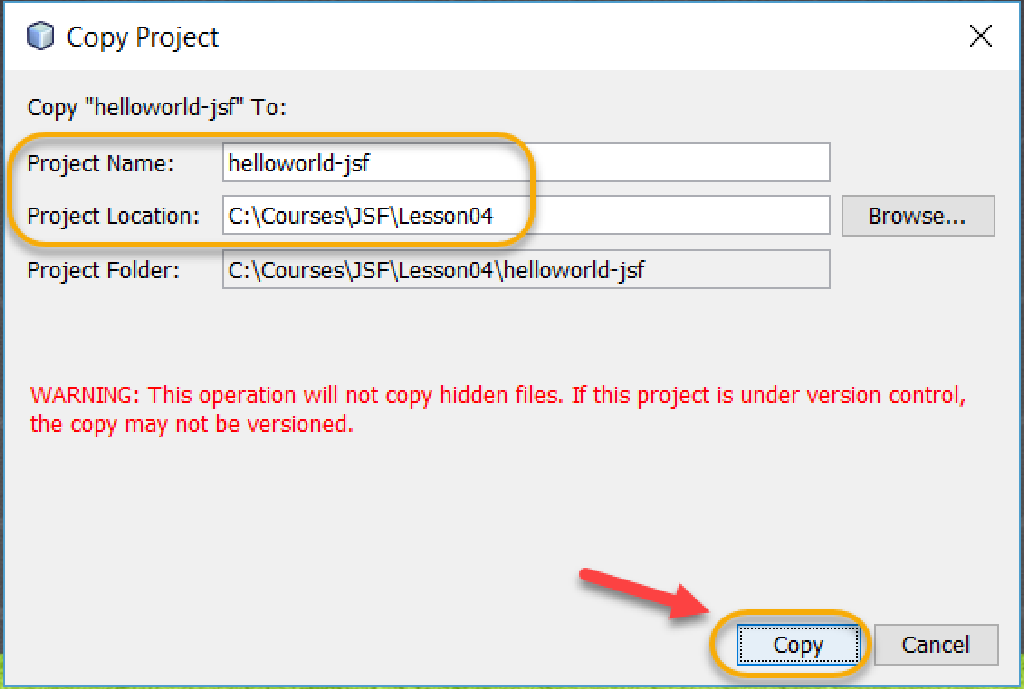
1. COPY THE PROJECT

We copy the project helloworld-jsf (the latest version):



1. COPY THE PROJECT

We copy the project helloworld-jsf (the latest version):



Copy Project

Copy "helloworld-jsf" To:

Project Name: helloworld-jsf

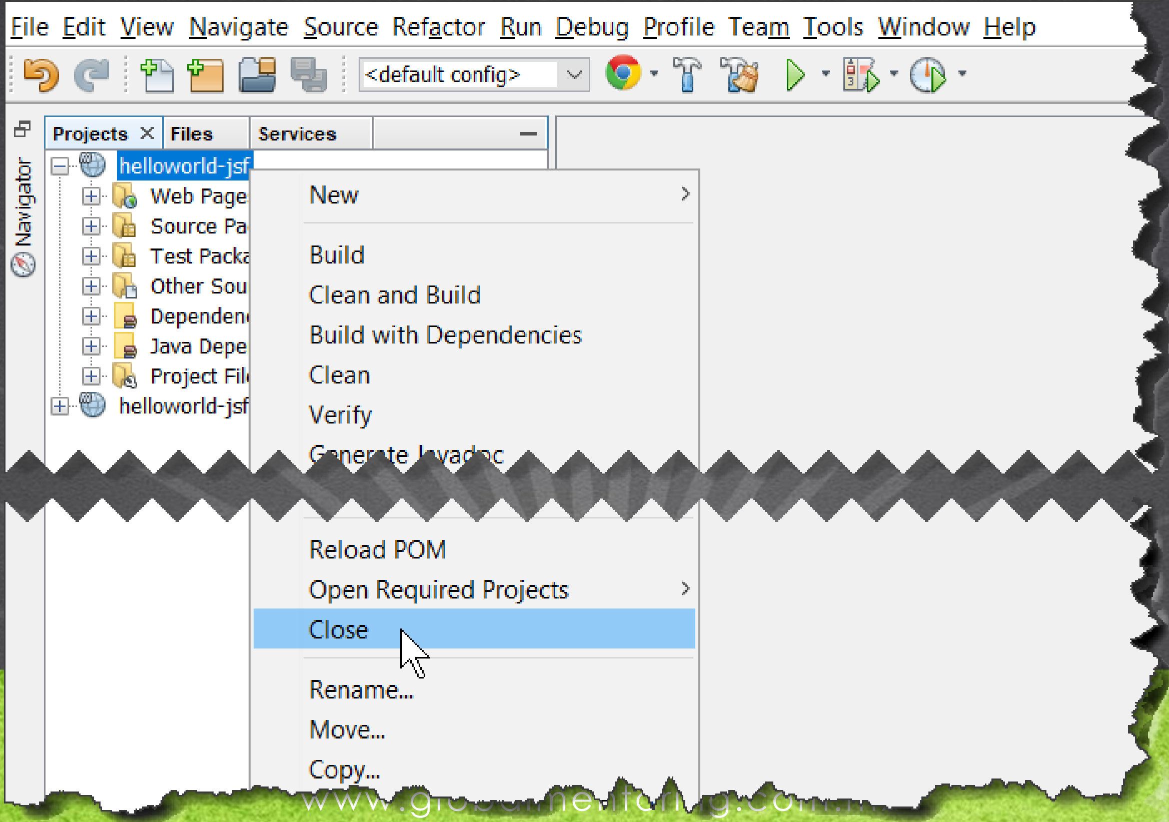
Project Location: C:\Courses\JSF\Lesson04

Project Folder: C:\Courses\JSF\Lesson04\helloworld-jsf

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

2. CLOSE THE PREVIOUS PROJECT

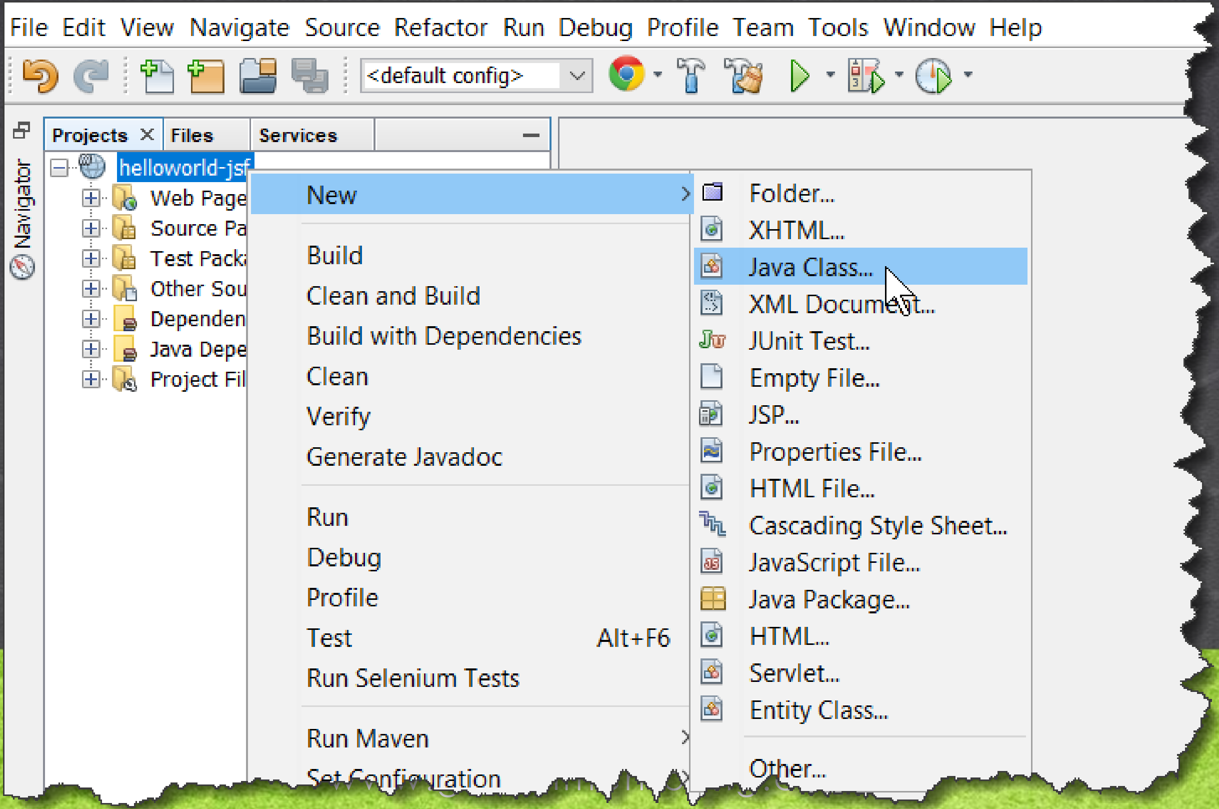
We closed the previous project, and left only the new one open:



www.globalmentoring.com.mx

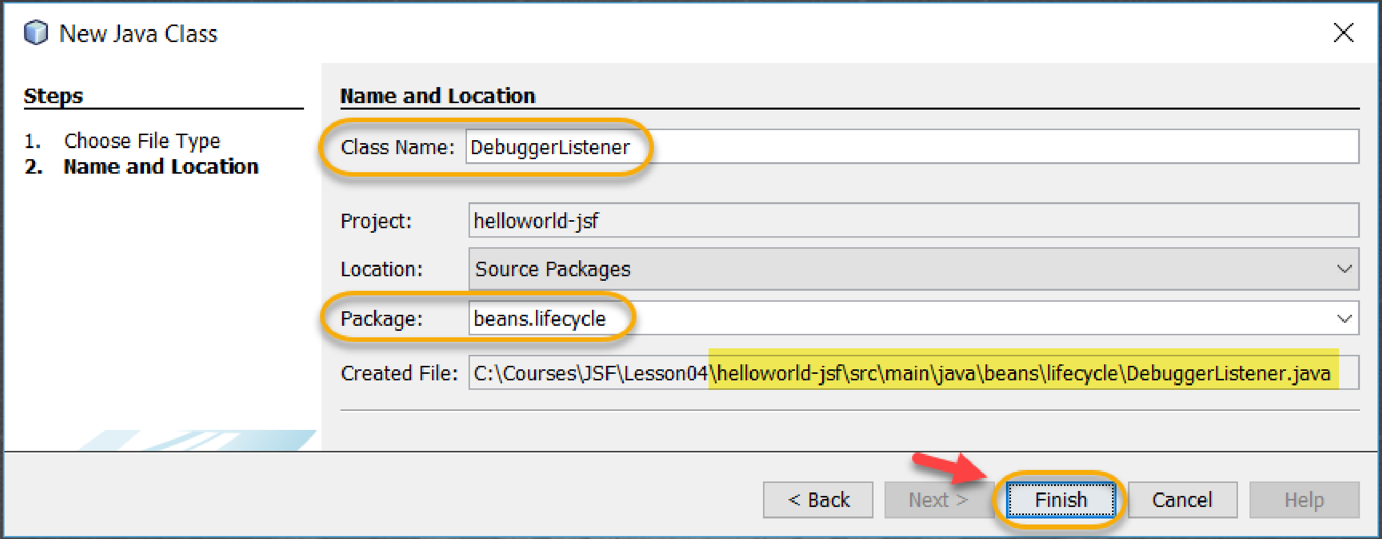
3. CREATE A NEW CLASS

We create the DebuggerListener.java class:



3. CREATE A NEW CLASS

We create the DebuggerListener.java class:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: DebuggerListener

Project: helloworld-jsf

Location: Source Packages

Package: beans.lifecycle

Created File: C:\Courses\JSF\Lesson04\helloworld-jsf\src\main\java\beans\lifecycle\DebuggerListener.java

4. MODIFY THE CODE

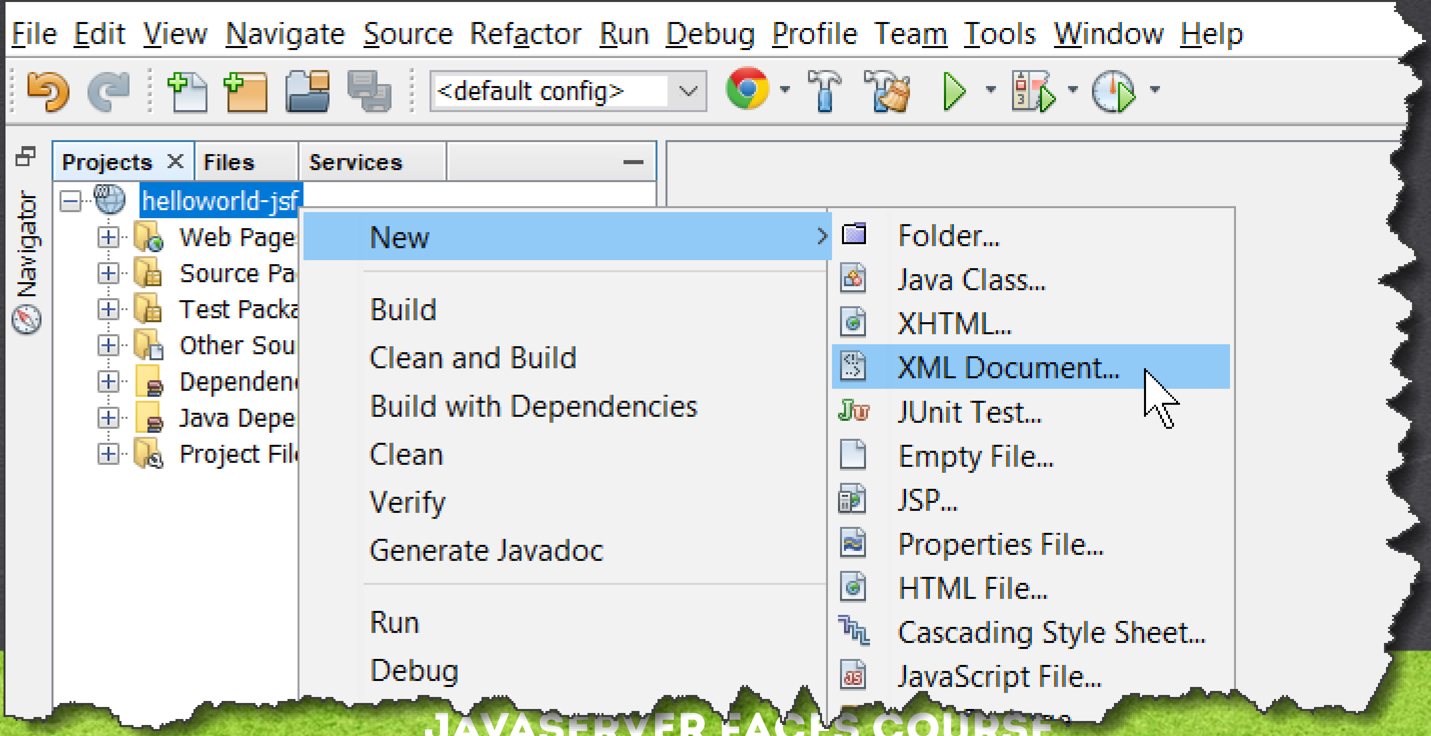
DebuggerListener.java:

Click to download

```java
package beans.lifecycle;

import javax.faces.event.PhaseEvent;
import javax.faces.event.PhaseId;
import org.apache.logging.log4j.*;

public class DebuggerListener implements javax.faces.event.PhaseListener {
    Logger log = LogManager.getRootLogger();

    @Override
    public void afterPhase(PhaseEvent phaseEvent) {

        if (log.isInfoEnabled()) {
            log.info("AFTER PHASE: " + phaseEvent.getPhaseId().toString());
        }
    }

    @Override
    public void beforePhase(PhaseEvent phaseEvent) {
        if (log.isInfoEnabled()) {
            log.info("BEFORE PHASE: " + phaseEvent.getPhaseId().toString());
        }
    }

    @Override
    public PhaseId getPhaseId() {
        return PhaseId.ANY_PHASE;
    }
}
```

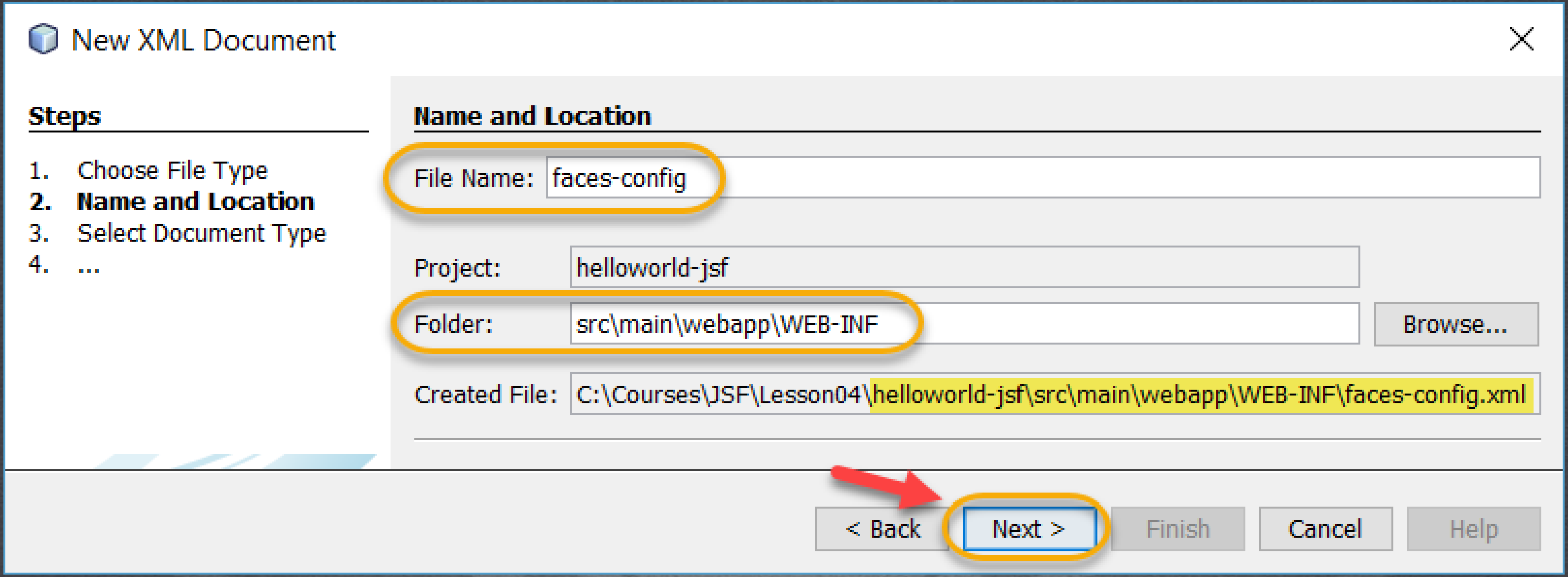
5. CREATE AN XML FILE

We created the faces-config.xml file:



5. CREATE AN XML FILE

We created the faces-config.xml file:



New XML Document

Steps

1. Choose File Type
2. **Name and Location**
3. Select Document Type
4. ...

Name and Location

File Name: faces-config

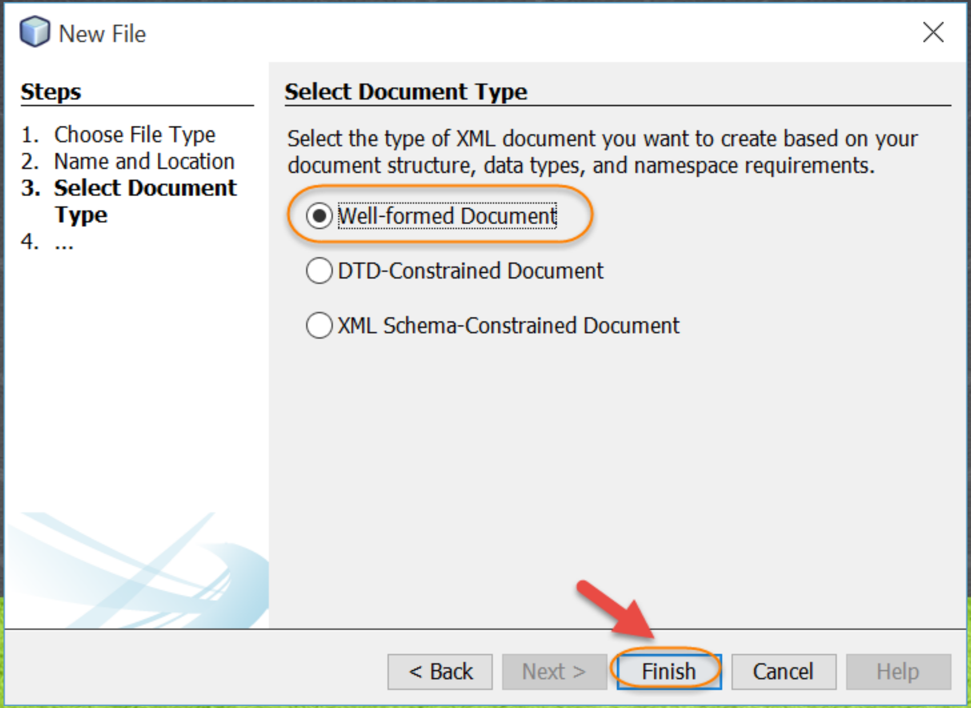
Project: helloworld-jsf

Folder: src\main\webapp\WEB-INF

Created File: C:\Courses\JSF\Lesson04\helloworld-jsf\src\main\webapp\WEB-INF\faces-config.xml

5. CREATE AN XML FILE

We created the faces-config.xml file. We select any option, it is not important since we are going to overwrite it:



New File

Steps

1. Choose File Type
2. Name and Location
3. **Select Document Type**
4. ...

Select Document Type

Select the type of XML document you want to create based on your document structure, data types, and namespace requirements.

● Well-formed Document

○ DTD-Constrained Document

○ XML Schema-Constrained Document

6. MODIFY THE CODE

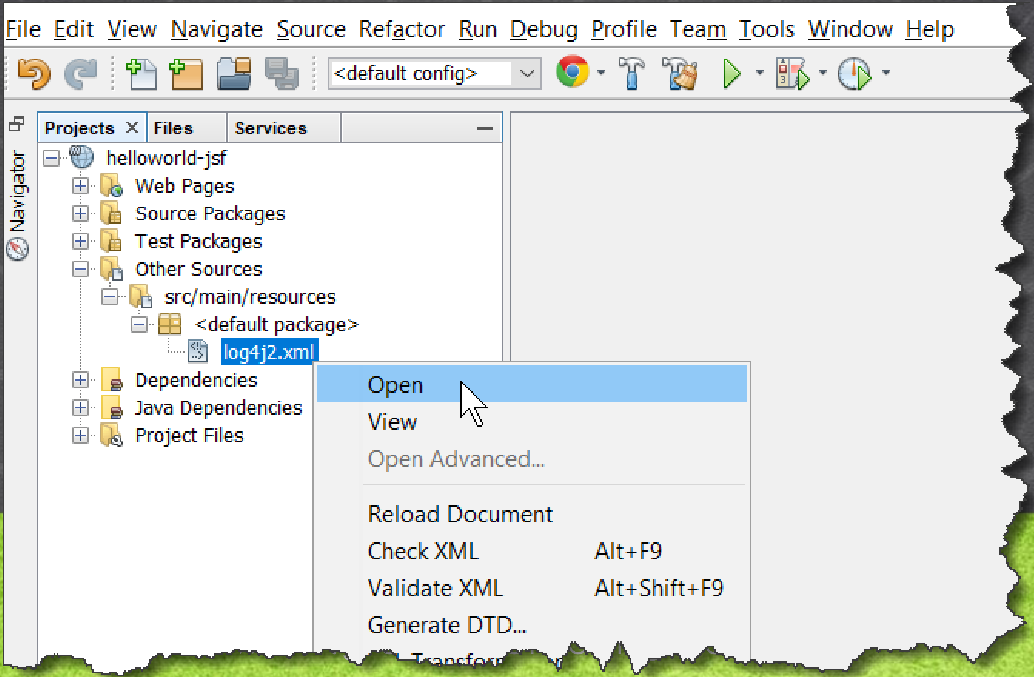
faces-config.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_3.xsd"
    version="2.3">

    <lifecycle>
        <phase-listener>
            beans.lifecycle.DebuggerListener
        </phase-listener>
    </lifecycle>

</faces-config>
```

7. MODIFY THE XML FILE

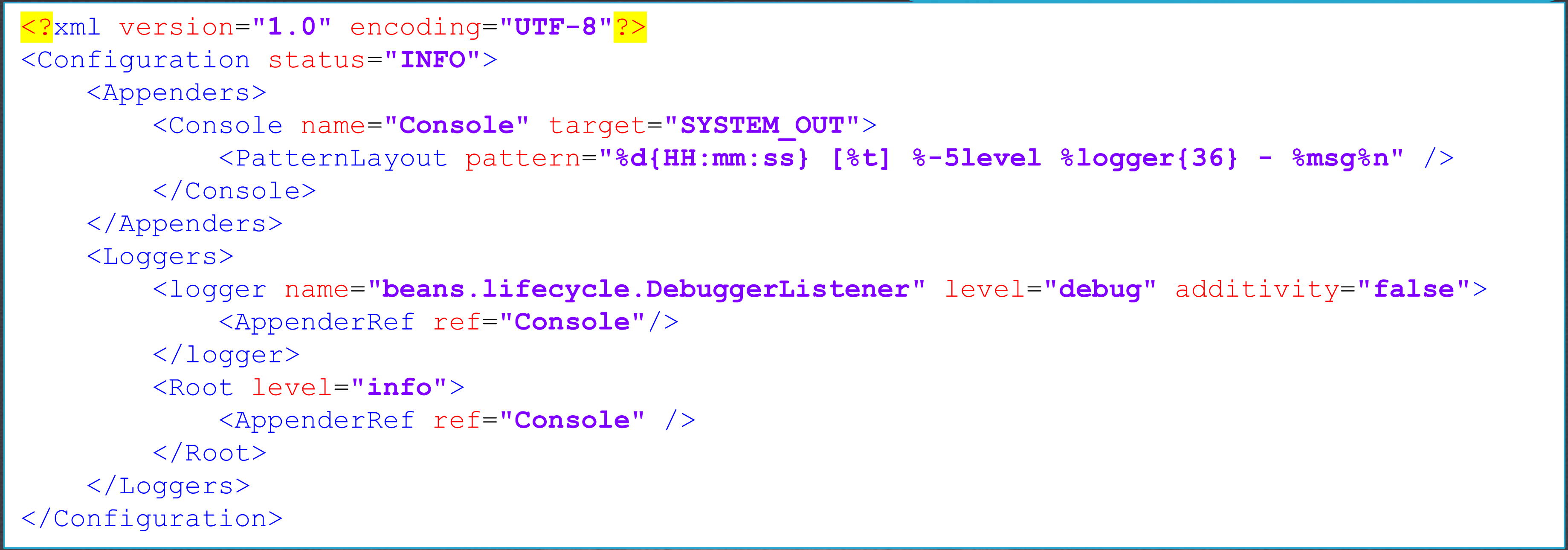
We modified the log4j2.xml file. We add the DebugListener class and put it in DEBUG mode so that we can observe the phases through which the JSF life cycle passes at the time of making a request, and receive the response from the web application. The file is in the following path:



7. MODIFY THE CODE

log4j2.xml:

Click to download



```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
        </Console>
    </Appenders>
    <Loggers>
        <logger name="beans.lifecycle.DebuggerListener" level="debug" additivity="false">
            <AppenderRef ref="Console"/>
        </logger>
        <Root level="info">
            <AppenderRef ref="Console" />
        </Root>
    </Loggers>
</Configuration>
```

8. MODIFY XML FILE

Modify the class Candidate.java to send to the server log if an object was created from the class constructor, and another message to know if the name property has been modified.

8. 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;

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

    public String getName() {
        return name;
    }

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

9. MODIFY XML FILE

Modify the VacantForm.java class to send to the server log if an object has been crated and also if it has been entered in the case of success or failure, according to the value entered in the input text.

9. MODIFY THE FILE

VacantForm.java:

Click to download

```java
package beans.backing;

import beans.model.Candidate;
import javax.inject.*;
import javax.enterprise.context.RequestScoped;
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;
    }
```

9. MODIFY THE FILE

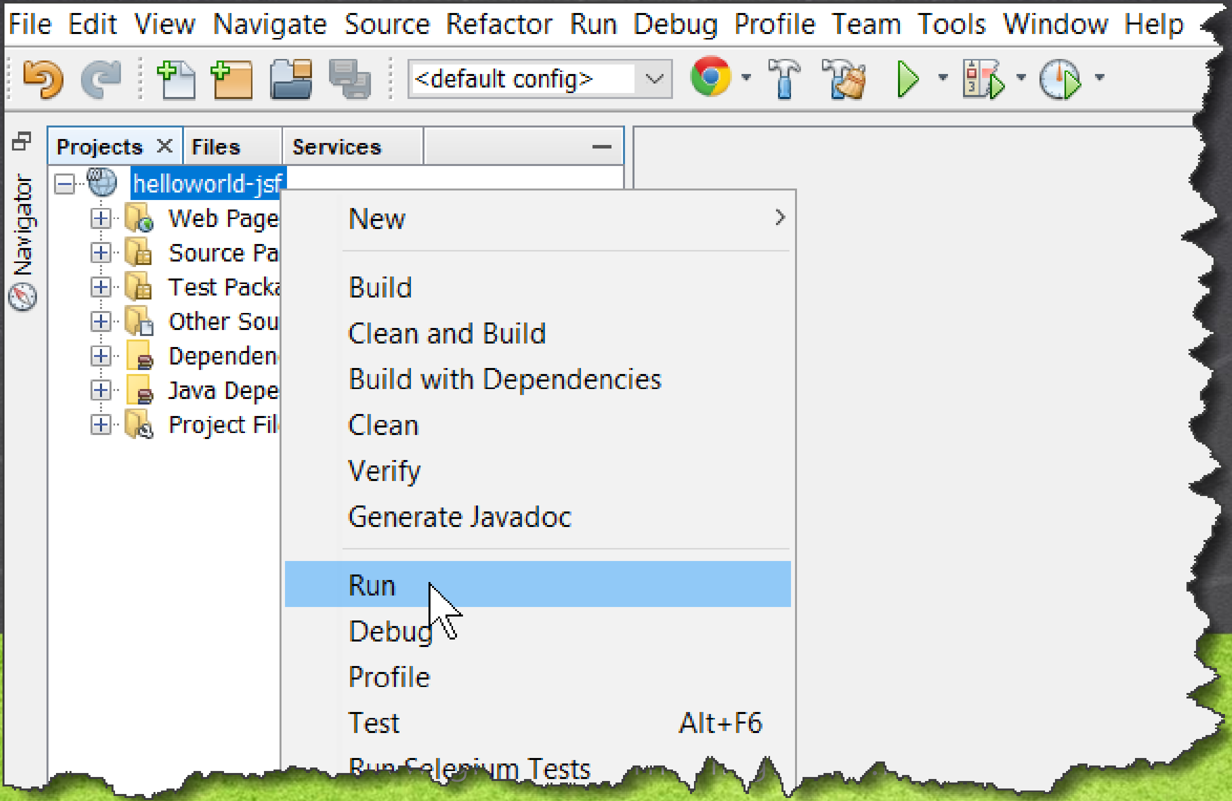
VacantForm.java:

Click to download

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

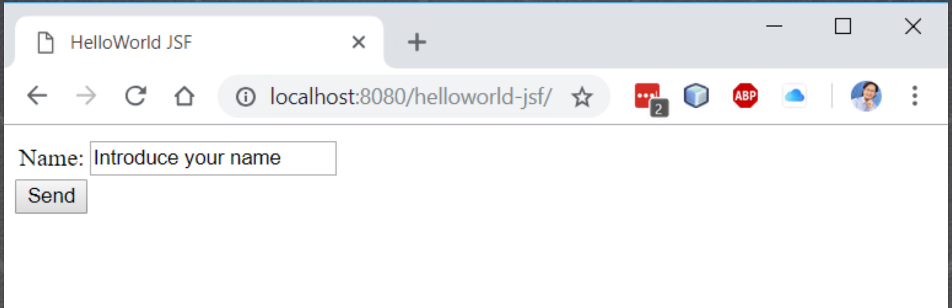
10. EXECUTE THE PROJECT

Execute the project:

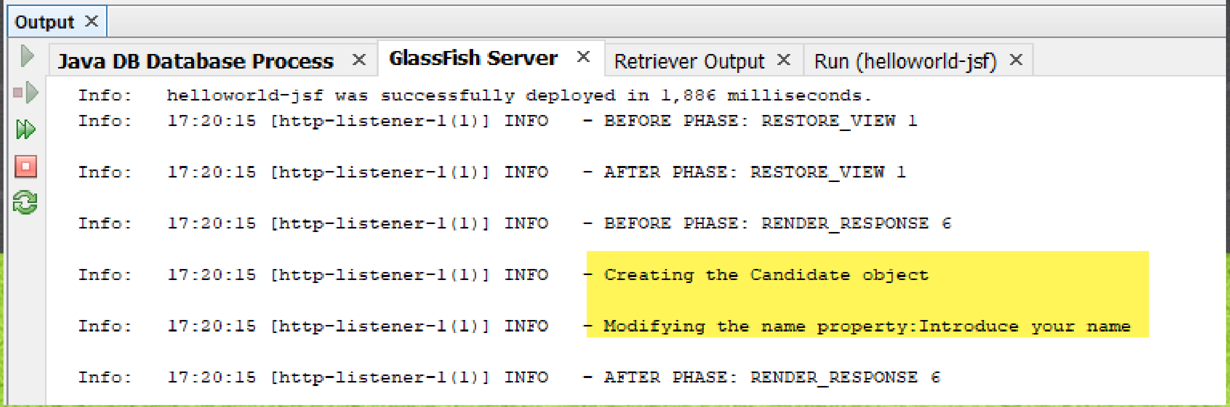


10. EXECUTE THE PROJECT

We execute our application and we will obtain the values of each of the phases of the JSF life cycle. This class will allow us to observe in which phase we are when processing the actions. When the initial page is shown, we obtain the following output:



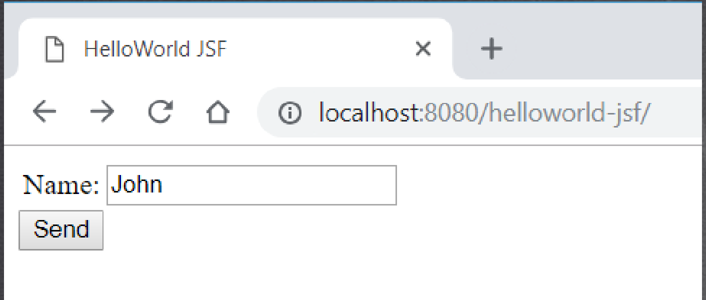
Name: Introduce your name



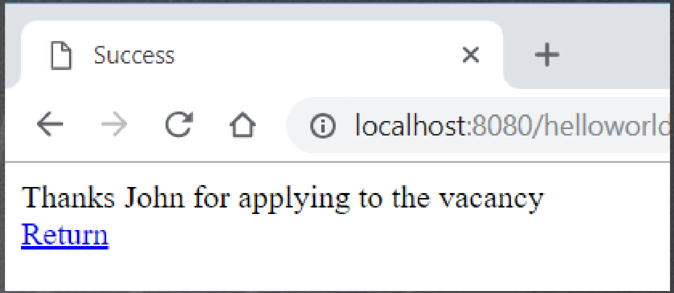
```
Info:   helloworld-jsf was successfully deployed in 1,886 milliseconds.
Info:   17:20:15 [http-listener-1(1)] INFO   - BEFORE PHASE: RESTORE_VIEW 1

Info:   17:20:15 [http-listener-1(1)] INFO   - AFTER PHASE: RESTORE_VIEW 1

Info:   17:20:15 [http-listener-1(1)] INFO   - BEFORE PHASE: RENDER_RESPONSE 6

Info:   17:20:15 [http-listener-1(1)] INFO   - Creating the Candidate object

Info:   17:20:15 [http-listener-1(1)] INFO   - Modifying the name property:Introduce your name

Info:   17:20:15 [http-listener-1(1)] INFO   - AFTER PHASE: RENDER_RESPONSE 6
```

10. EXECUTE THE PROJECT

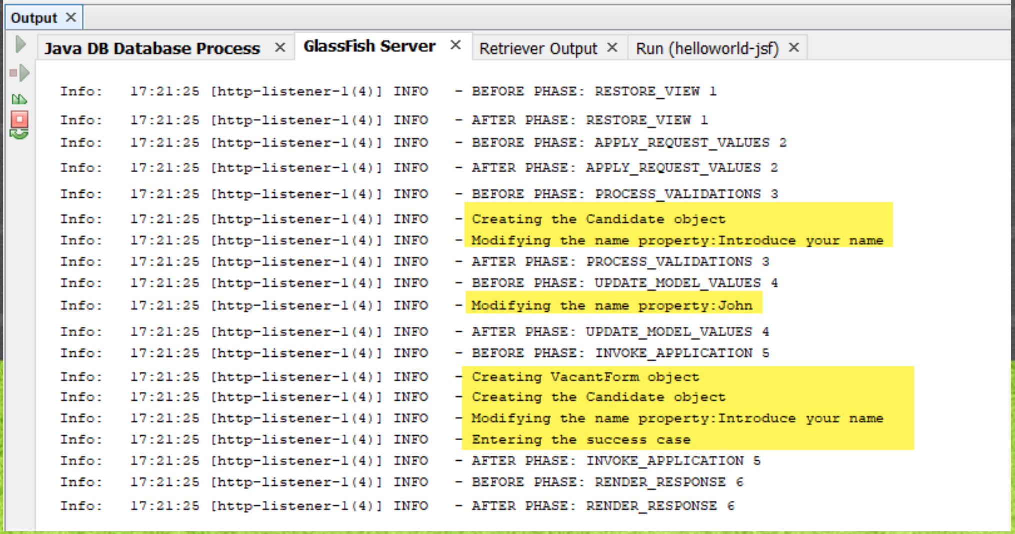
By providing the value of John in the text field we see the following output :



Name: John



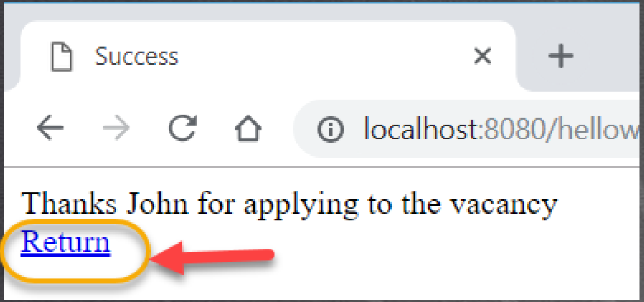
Thanks John for applying to the vacancy
Return



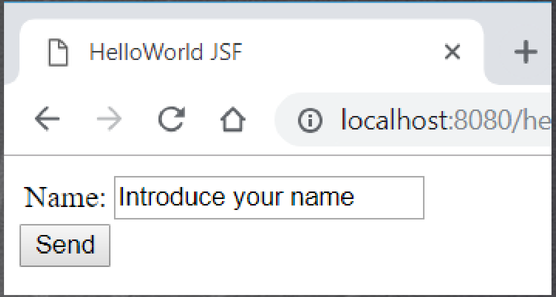
```
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: RESTORE_VIEW 1
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: RESTORE_VIEW 1
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: APPLY_REQUEST_VALUES 2
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: PROCESS_VALIDATIONS 3
Info:   17:21:25 [http-listener-1(4)] INFO  - Creating the Candidate object
Info:   17:21:25 [http-listener-1(4)] INFO  - Modifying the name property:Introduce your name
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: PROCESS_VALIDATIONS 3
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Info:   17:21:25 [http-listener-1(4)] INFO  - Modifying the name property:John
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: UPDATE_MODEL_VALUES 4
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: INVOKE_APPLICATION 5
Info:   17:21:25 [http-listener-1(4)] INFO  - Creating VacantForm object
Info:   17:21:25 [http-listener-1(4)] INFO  - Creating the Candidate object
Info:   17:21:25 [http-listener-1(4)] INFO  - Modifying the name property:Introduce your name
Info:   17:21:25 [http-listener-1(4)] INFO  - Entering the success case
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: INVOKE_APPLICATION 5
Info:   17:21:25 [http-listener-1(4)] INFO  - BEFORE PHASE: RENDER_RESPONSE 6
Info:   17:21:25 [http-listener-1(4)] INFO  - AFTER PHASE: RENDER_RESPONSE 6
```

10. EXECUTE THE PROJECT

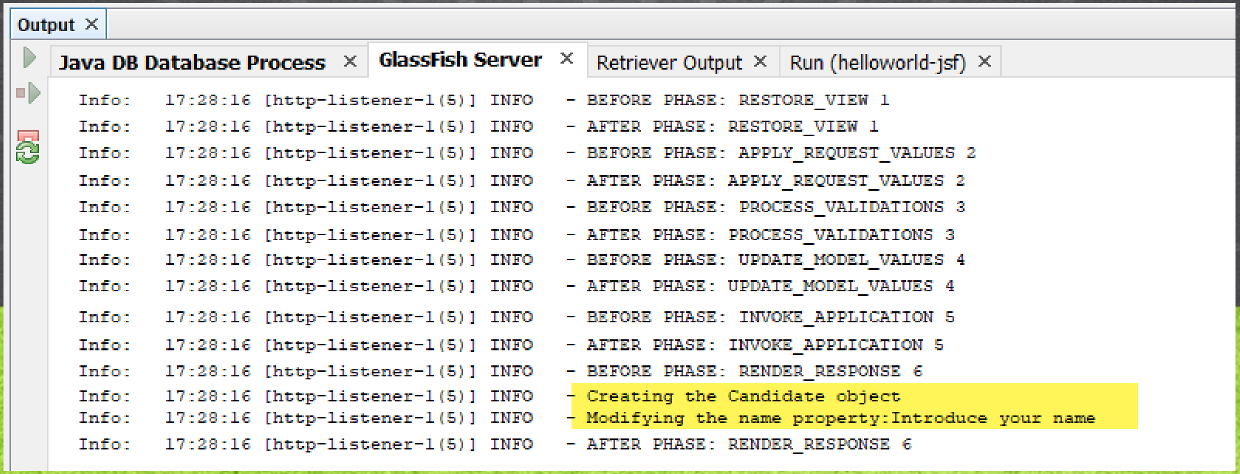
When pressing the return button we observe the following output, which is different from when the page was requested for the first time when starting the application



Thanks John for applying to the vacancy
Return



Name: Introduce your name
Send



```
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: RESTORE_VIEW 1
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: RESTORE_VIEW 1
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: APPLY_REQUEST_VALUES 2
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: PROCESS_VALIDATIONS 3
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: PROCESS_VALIDATIONS 3
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: UPDATE_MODEL_VALUES 4
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: INVOKE_APPLICATION 5
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: INVOKE_APPLICATION 5
Info:   17:28:16 [http-listener-1(5)] INFO  - BEFORE PHASE: RENDER_RESPONSE 6
Info:   17:28:16 [http-listener-1(5)] INFO  - Creating the Candidate object
Info:   17:28:16 [http-listener-1(5)] INFO  - Modifying the name property:Introduce your name
Info:   17:28:16 [http-listener-1(5)] INFO  - AFTER PHASE: RENDER_RESPONSE 6
```

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

EXERCISE CONCLUSION

With this exercise we have put into practice the life cycle in JSF.

We did several tests and observed how in each one of them we obtain certain values in different stages of the life cycle of JSF. This kind of DebuggerListener will help us to debug our application and be able to detect errors more easily and in which phase or stage the error is being caused.

ONLINE COURSE

JAVASERVER FACES (JSF)

By: Eng. Ubaldo Acosta

Global Mentoring

JAVASERVER FACES COURSE

www.globalmentoring.com.mx