EXERCISE

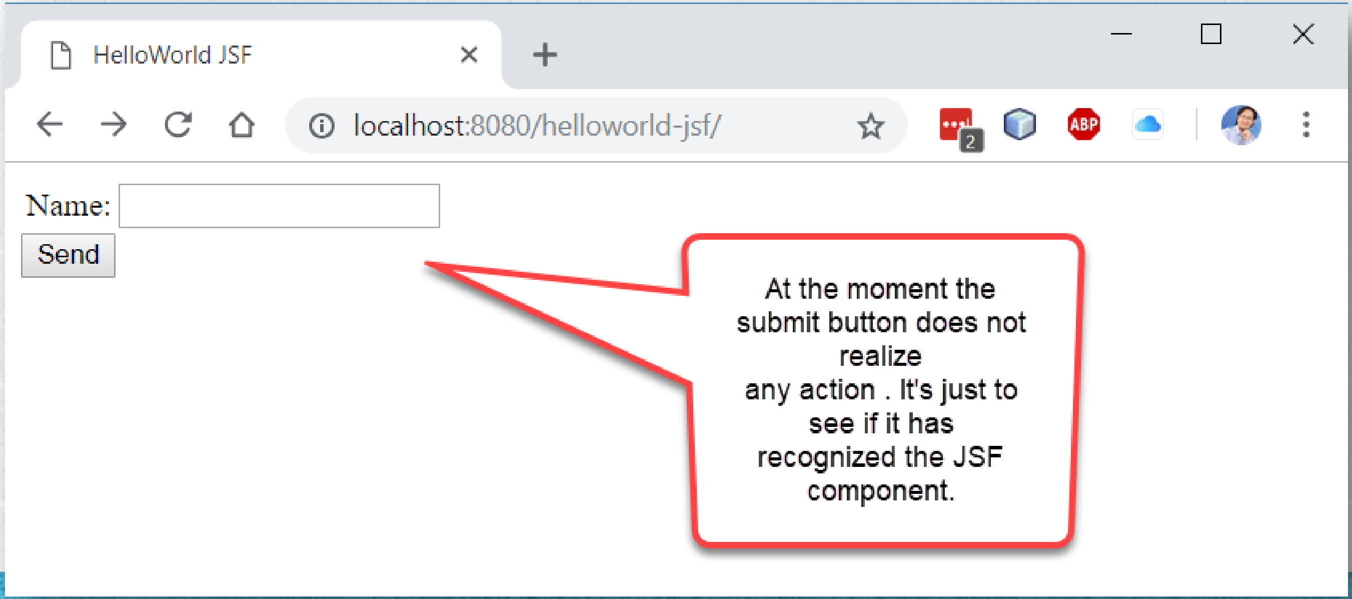
HELLO WORLD WITH JAVASERVER FACES

Global Mentoring

Experiencia y Conocimiento para tu vida

EXERCISE OBJECTIVE

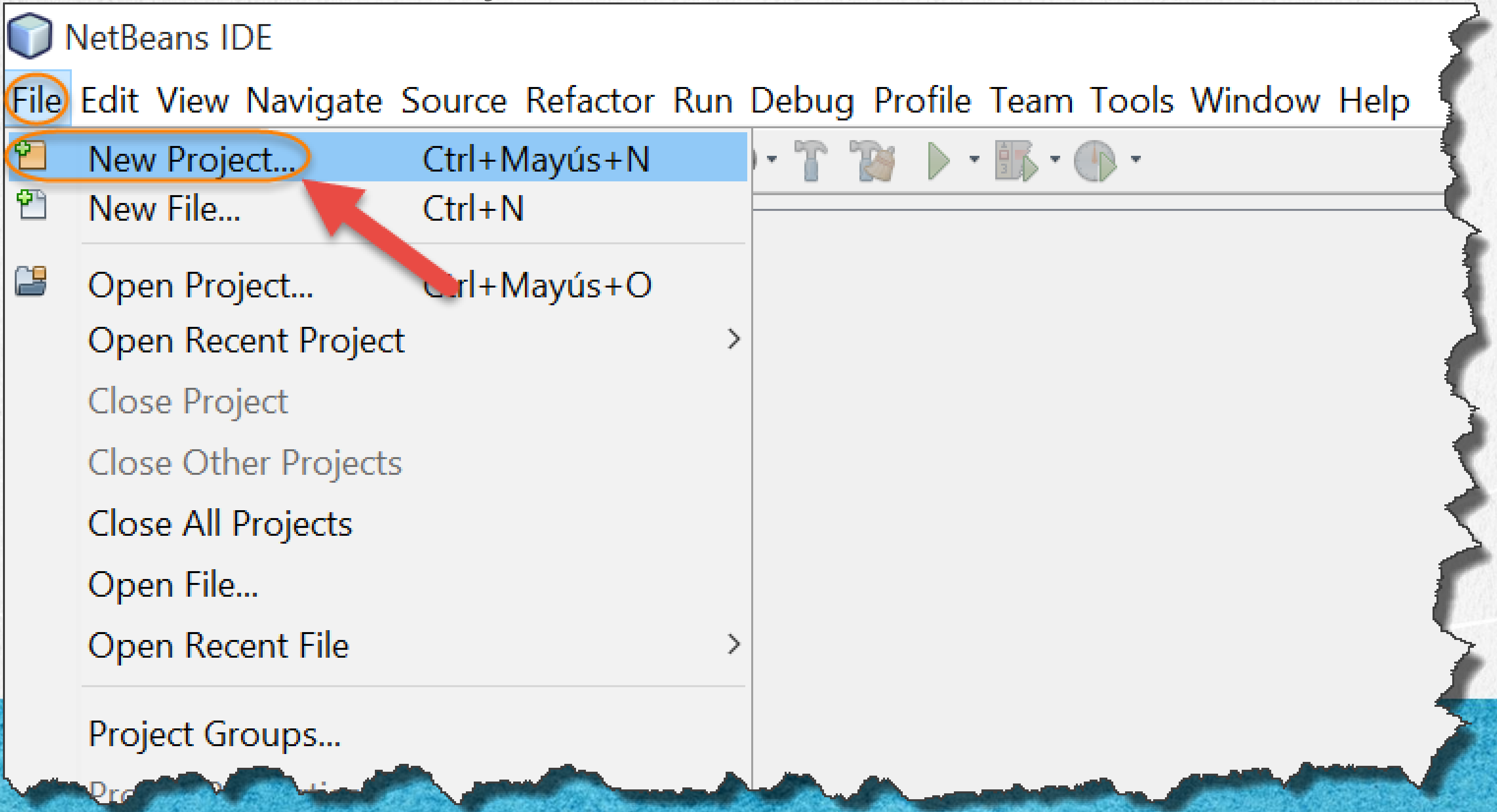
- The objective of the exercise is to set up a HelloWorld project with JSF. We will rely on Maven for the creation of the project. The result should be similar to the following:



At the moment the submit button does not realize any action . It's just to see if it has recognized the JSF component.

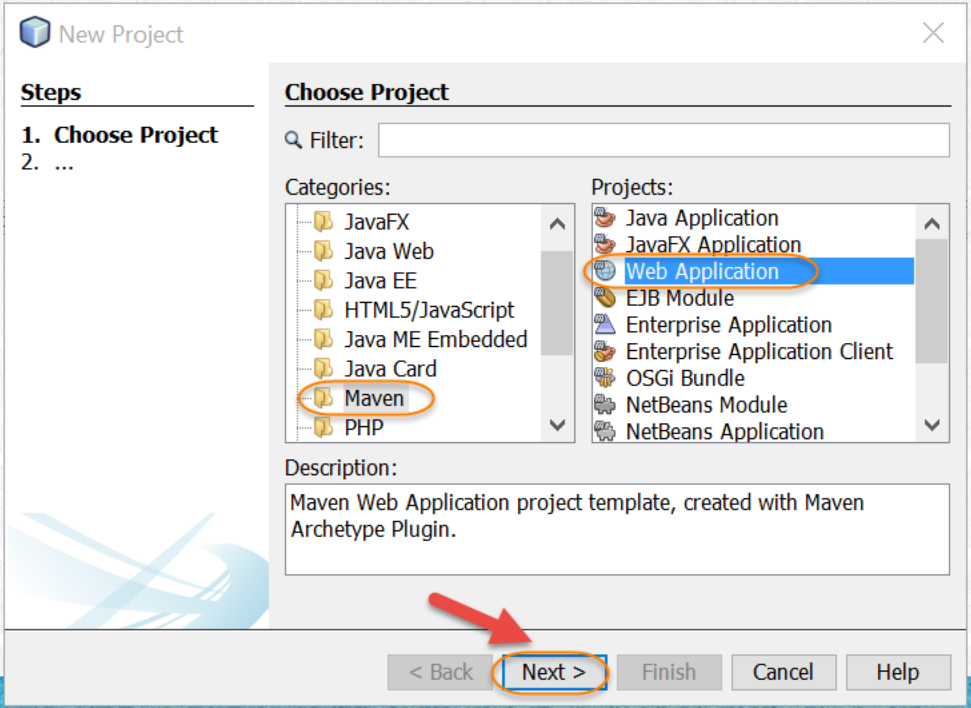
1. CREATE A NEW PROJECT

Create a new Java Web Project with Maven:



1. CREATE A NEW PROJECT

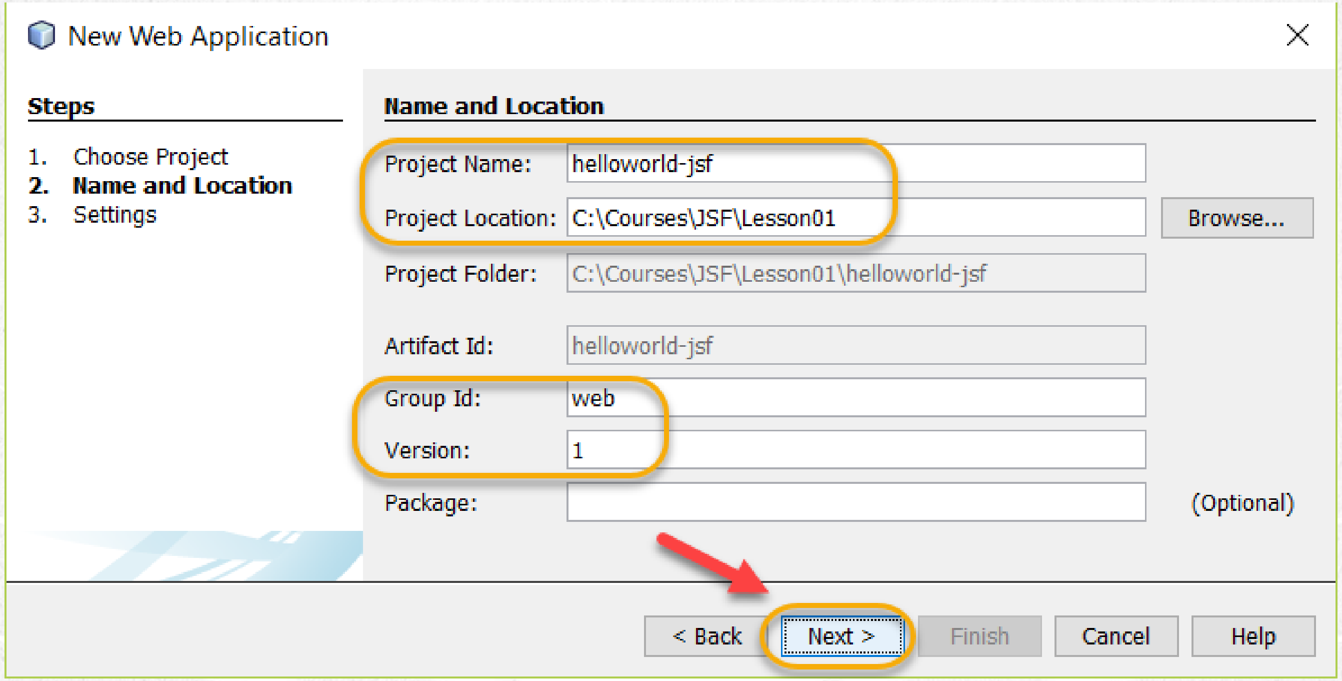
Create a new Java Web Project with Maven:



Maven Web Application project template, created with Maven Archetype Plugin.

1. CREATE A NEW PROJECT

We write the following values:



New Web Application

Steps

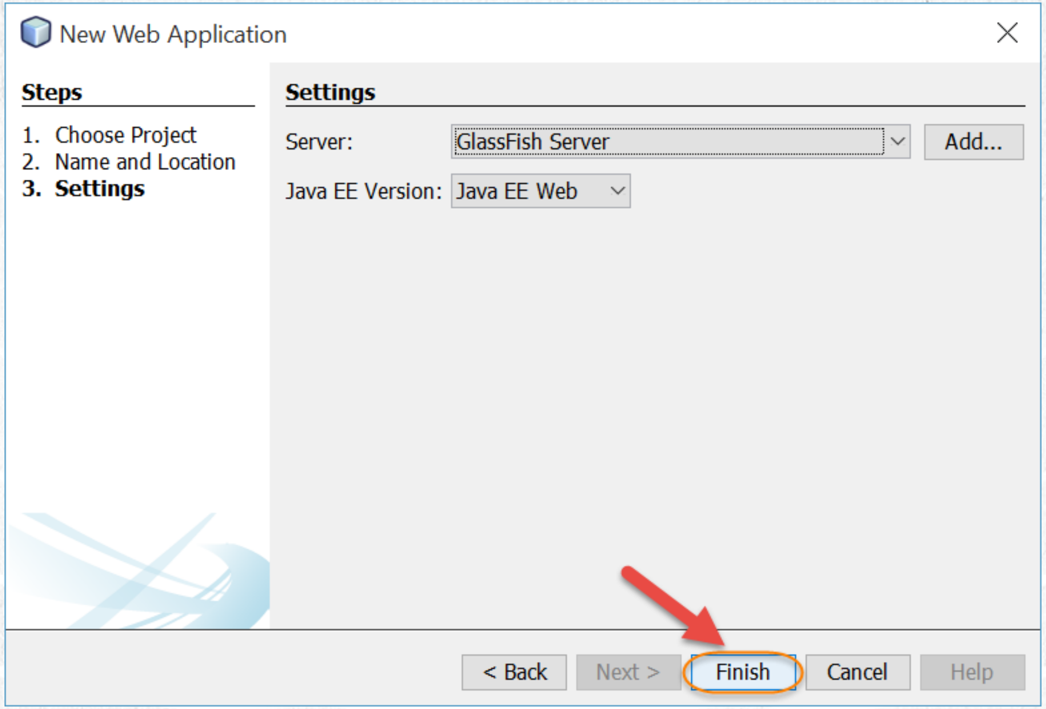
1. Choose Project
2. **Name and Location**
3. Settings

Name and Location

Project Name:	helloworld-jsf
Project Location:	C:\Courses\JSF\Lesson01
Project Folder:	C:\Courses\JSF\Lesson01\helloworld-jsf
Artifact Id:	helloworld-jsf
Group Id:	web
Version:	1
Package:	(Optional)

1. CREATE A NEW PROJECT

Select the following options:


New Web Application

Steps
1. Choose Project
2. Name and Location
3. **Settings**

Settings

Server: GlassFish Server

Java EE Version: Java EE Web

2. ADD JSF LIBRARIES

We are going to use Maven to manage the dependencies that we are going to use in the project. The javaee-web library already includes the necessary dependencies for JSF.

We add the following .jar libraries to the pom.xml file:

- javaee-web
- log4j

2. MODIFY THE CODE

pom.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>web</groupId>
    <artifactId>helloworld-jsf</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>helloworld-jsf</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <log4j.version>2.11.1</log4j.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
```

2. MODIFY THE CODE

pom.xml:

Click to download

```
        <!-- Log4j -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>${log4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>${log4j.version}</version>
        </dependency>
    </dependencies>
```

2. MODIFY THE CODE

pom.xml:

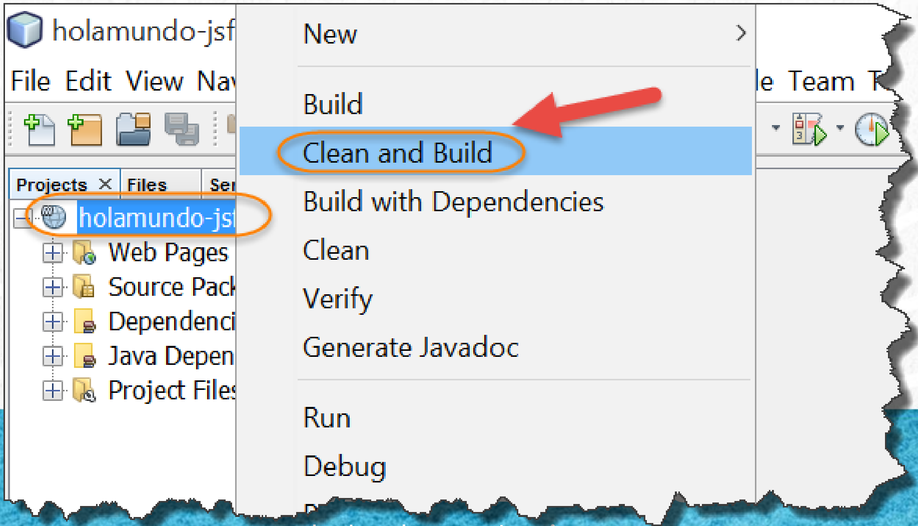
Click to download

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.3</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

3. MAVEN REPOSITORY UPDATE)

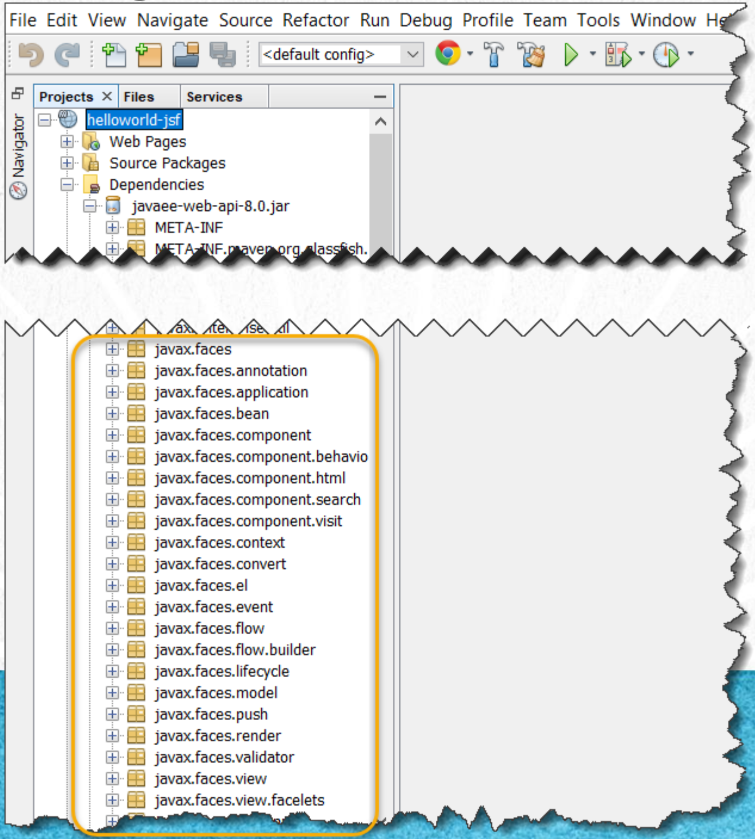
To update the Maven repository, which will be responsible for administering the .jar libraries of our project, it is enough to make clean & build our project

Note: If for some reason the repository is not updated, disable the antivirus or verify if you have a proxy configuration.



4. REVIEW OF THE DEPENDENCIES

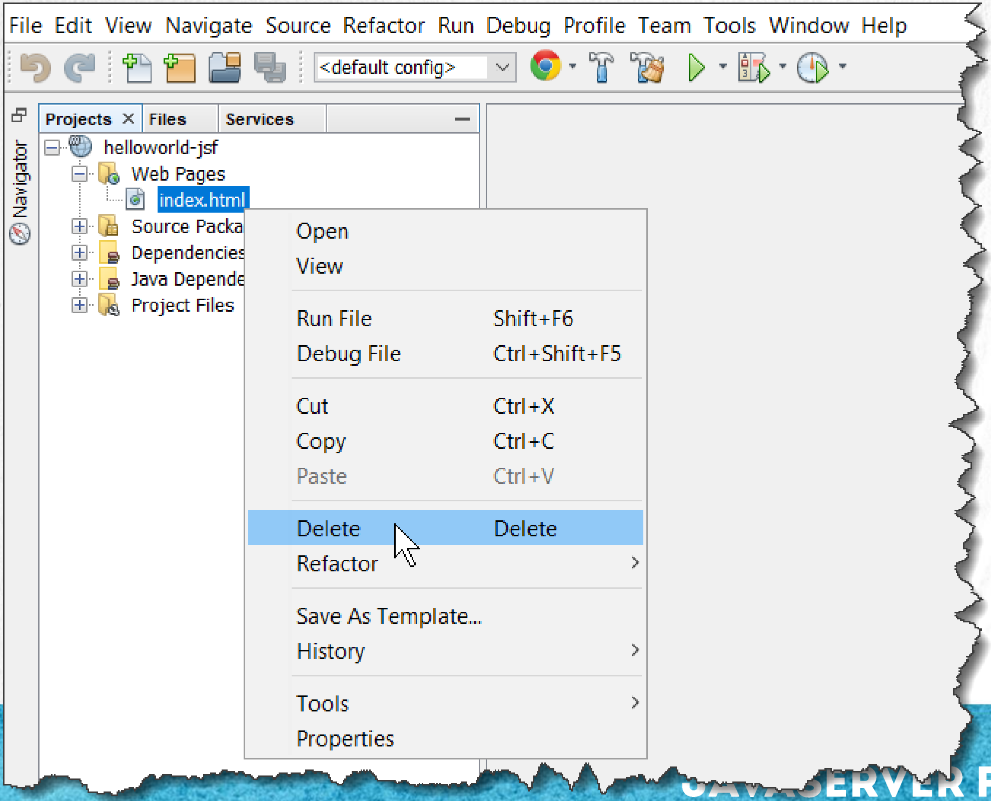
We check that the project has the javaee-web library and that we observe that it has the javax.faces package classes:

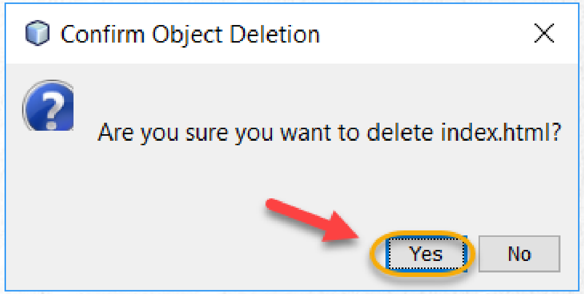

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window He
<default config>
Navigator
Projects
Files
Services

helloworld-jsf
Web Pages
Source Packages
Dependencies
javaee-web-api-8.0.jar
META-INF
META-INF.maven.org.glassfish.
javax.faces
javax.faces.annotation
javax.faces.application
javax.faces.bean
javax.faces.component
javax.faces.component.behavio
javax.faces.component.html
javax.faces.component.search
javax.faces.component.visit
javax.faces.context
javax.faces.convert
javax.faces.el
javax.faces.event
javax.faces.flow
javax.faces.flow.builder
javax.faces.lifecycle
javax.faces.model
javax.faces.push
javax.faces.render
javax.faces.validator
javax.faces.view
javax.faces.view.facelets

5. WE REMOVED THE HTML FILE

If it exists, we remove the index.html file:

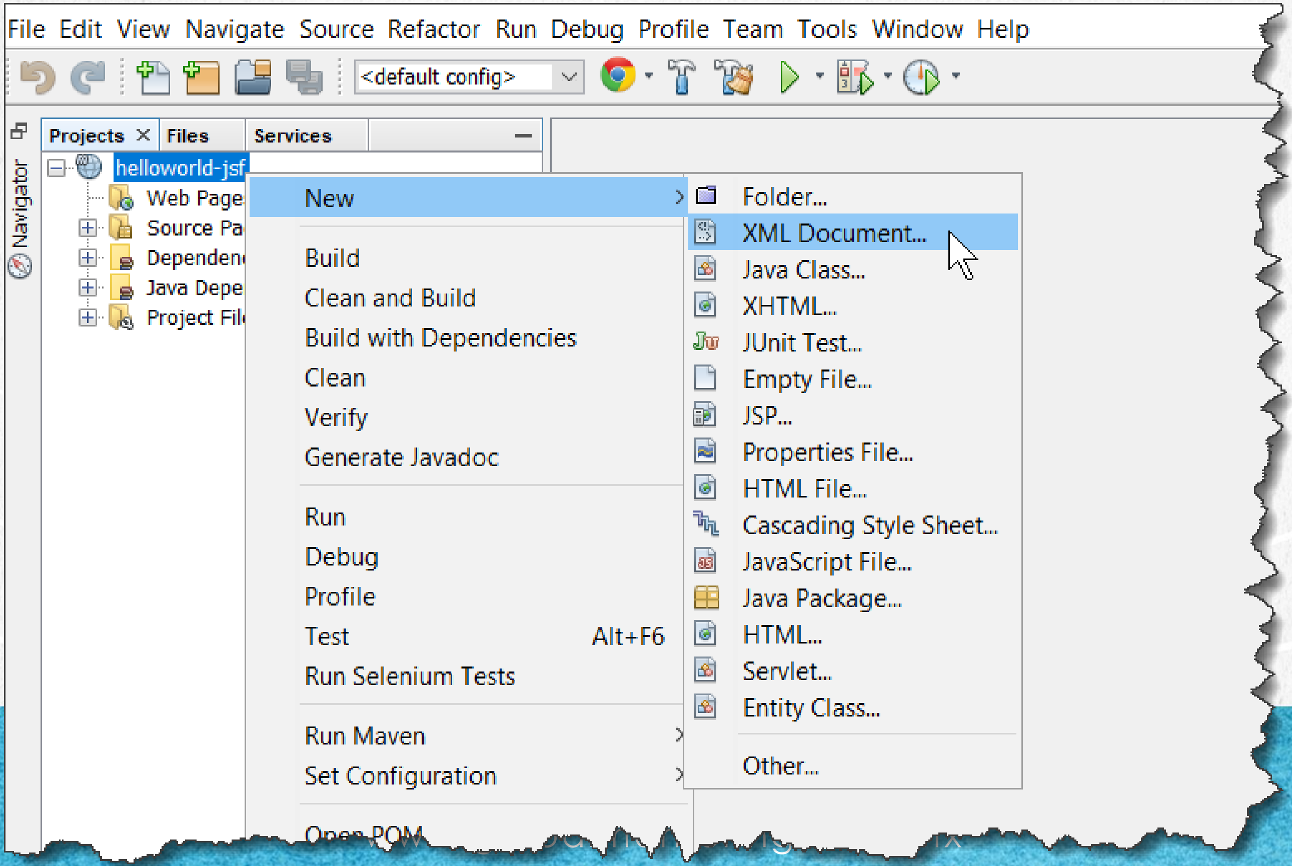




Are you sure you want to delete index.html?

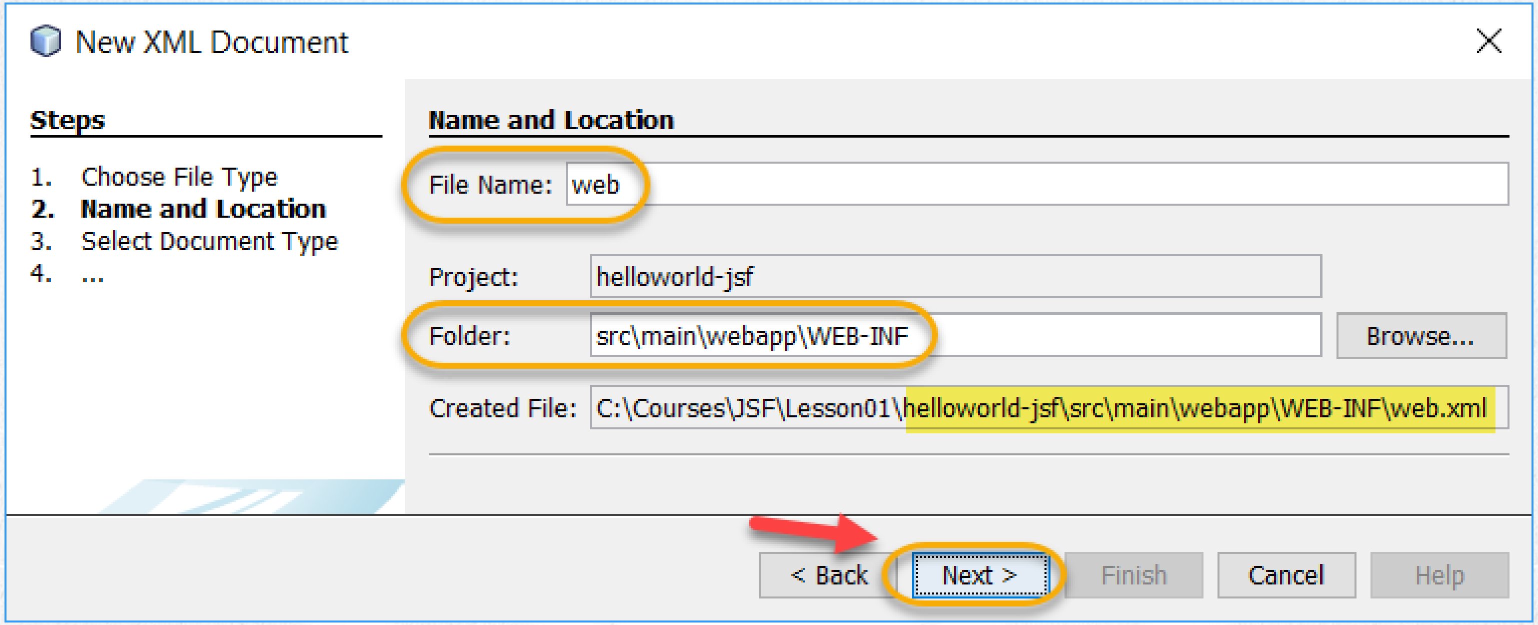
6. We create an XML file

We create the web.xml file:



6. WE CREATE AN XML FILE

We create the web.xml file inside the WEB-INF folder as shown:



New XML Document

Steps

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

Name and Location

File Name: web

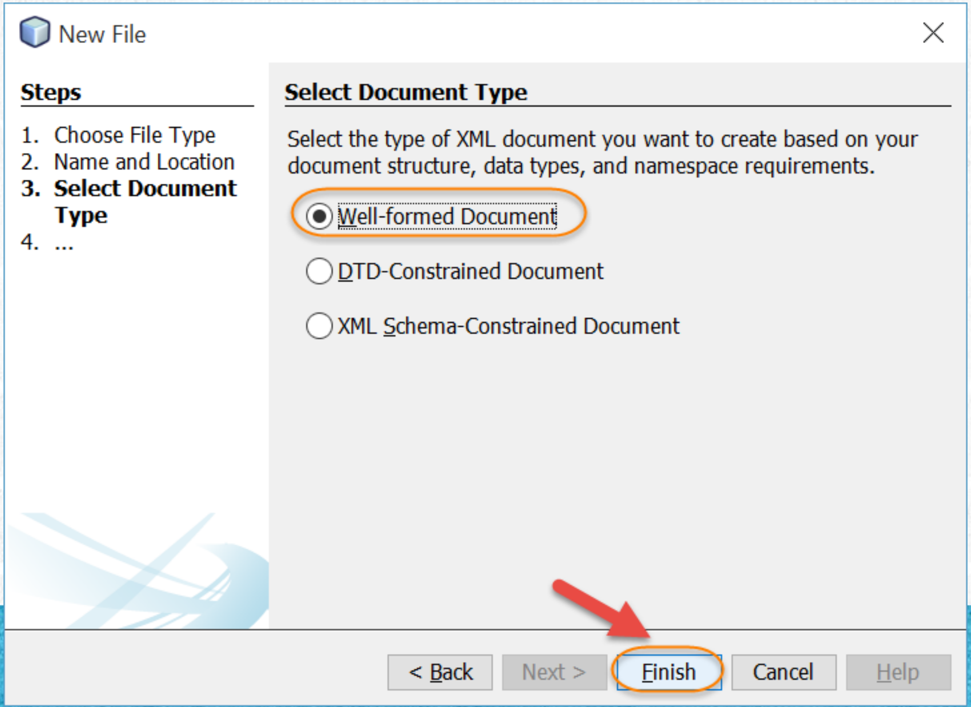
Project: helloworld-jsf

Folder: src\main\webapp\WEB-INF

Created File: C:\Courses\JSF\Lesson01\helloworld-jsf\src\main\webapp\WEB-INF\web.xml

6. CREATE AN XML FILE

We create the web.xml file. We select any option, it is not important since we will overwrite the contents of the xml file:



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

7. MODIFY THE CODE

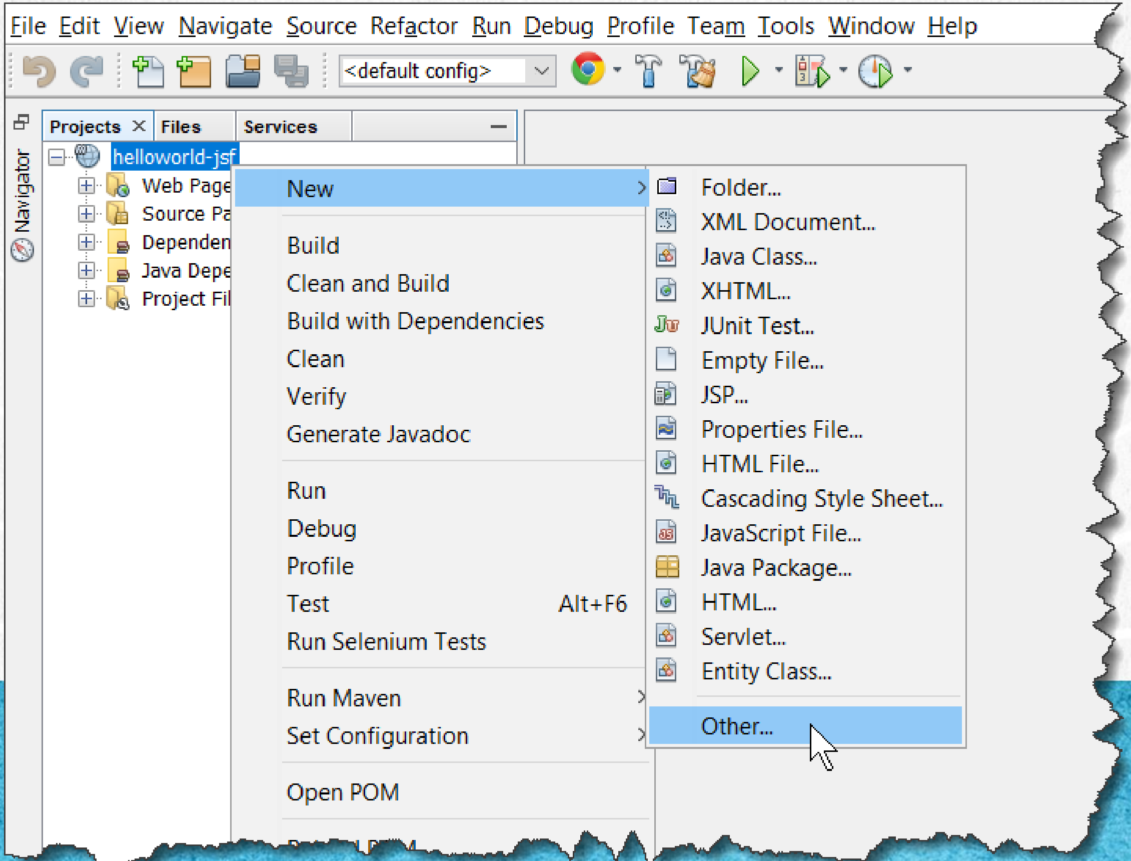
web.xml:

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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 webapp_4_0.xsd"
         version="4.0">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

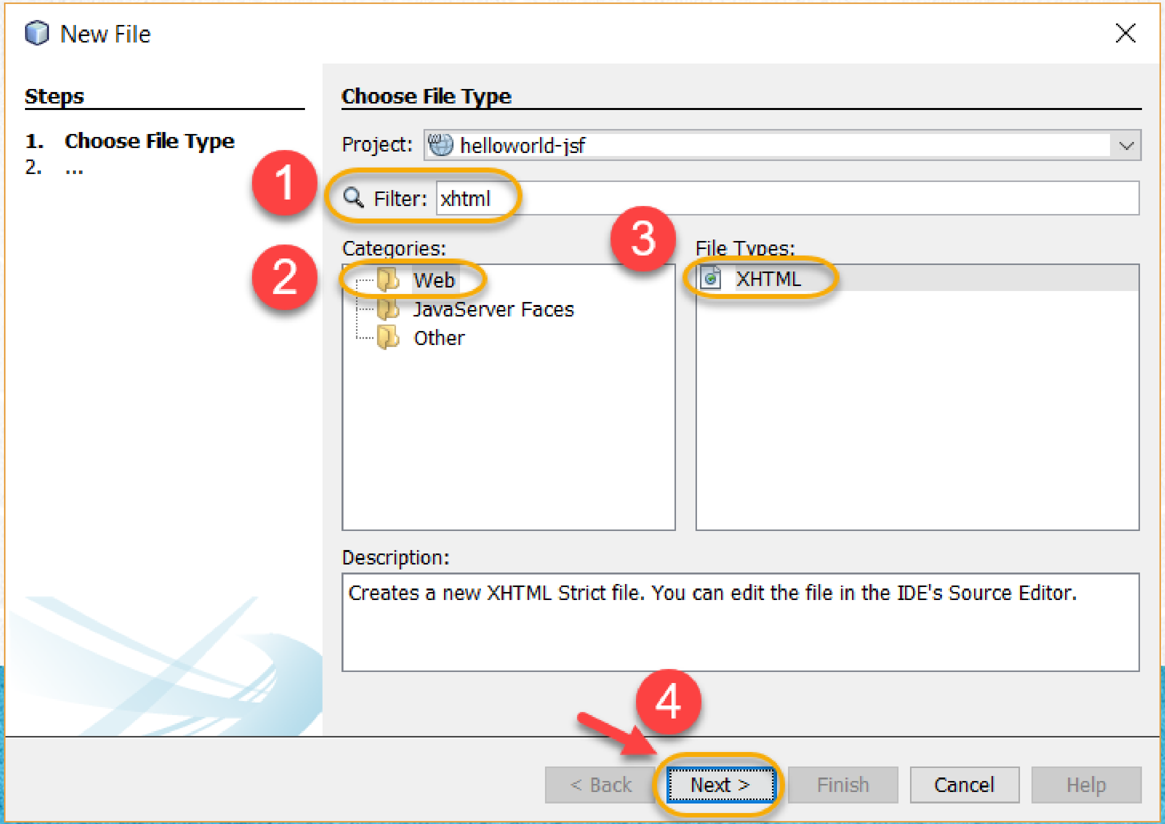
8. Create an XHTML file

We create the index.xhtml file:



8. CREATE AN XHTML FILE

We create the index.xhtml file:



Choose File Type

Project: helloworld-jsf

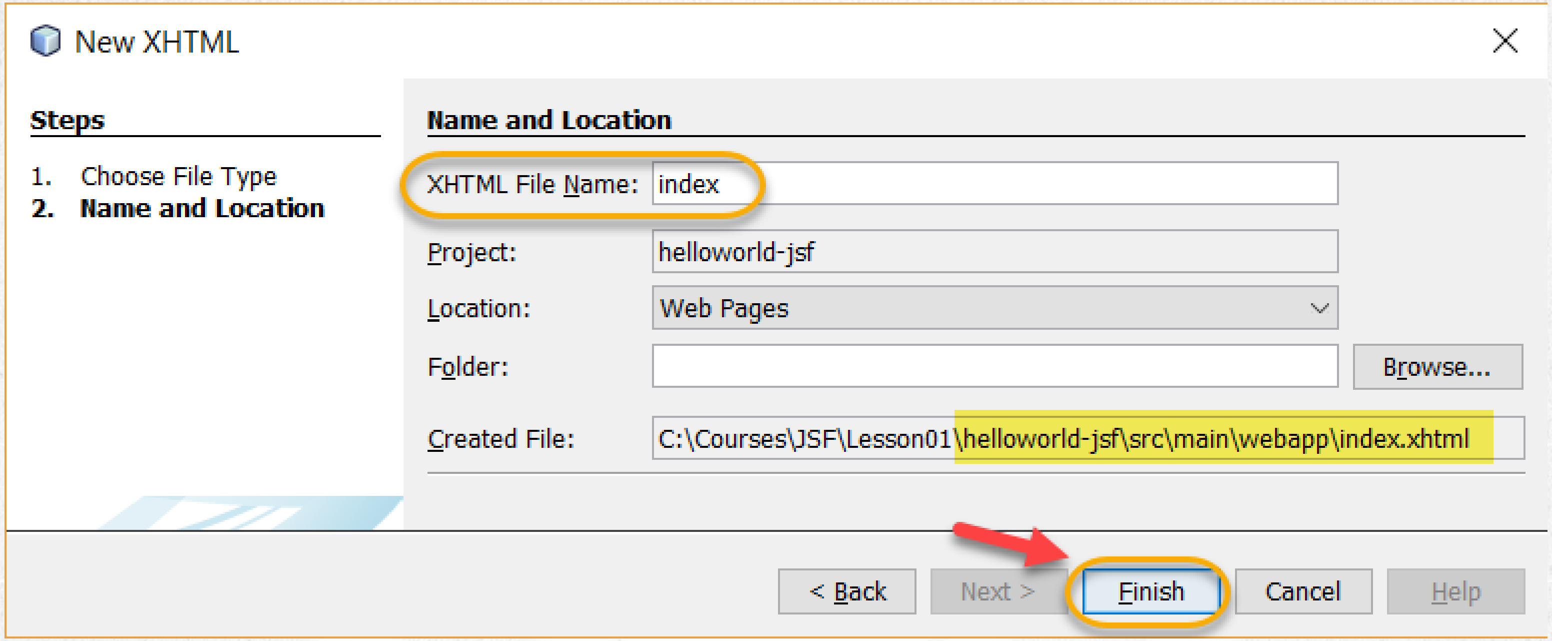
Filter: xhtml

Description:

Creates a new XHTML Strict file. You can edit the file in the IDE's Source Editor.

8. CREATE AN XHTML FILE

We create the index.xhtml file:



XHTML File Name: index

Project: helloworld-jsf

Location: Web Pages

Folder:

Created File: C:\Courses\JSF\Lesson01\helloworld-jsf\src\main\webapp\index.xhtml

9. MODIFY THE CODE

index.xhtml:

Click to download

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<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" /></td>
                    <td><h:message for="name" /></td>
                </tr>
            </table>
            <h:commandButton value="Send" />
        </h:form>
    </h:body>
</html>
```

10. CREATE AN XML FILE

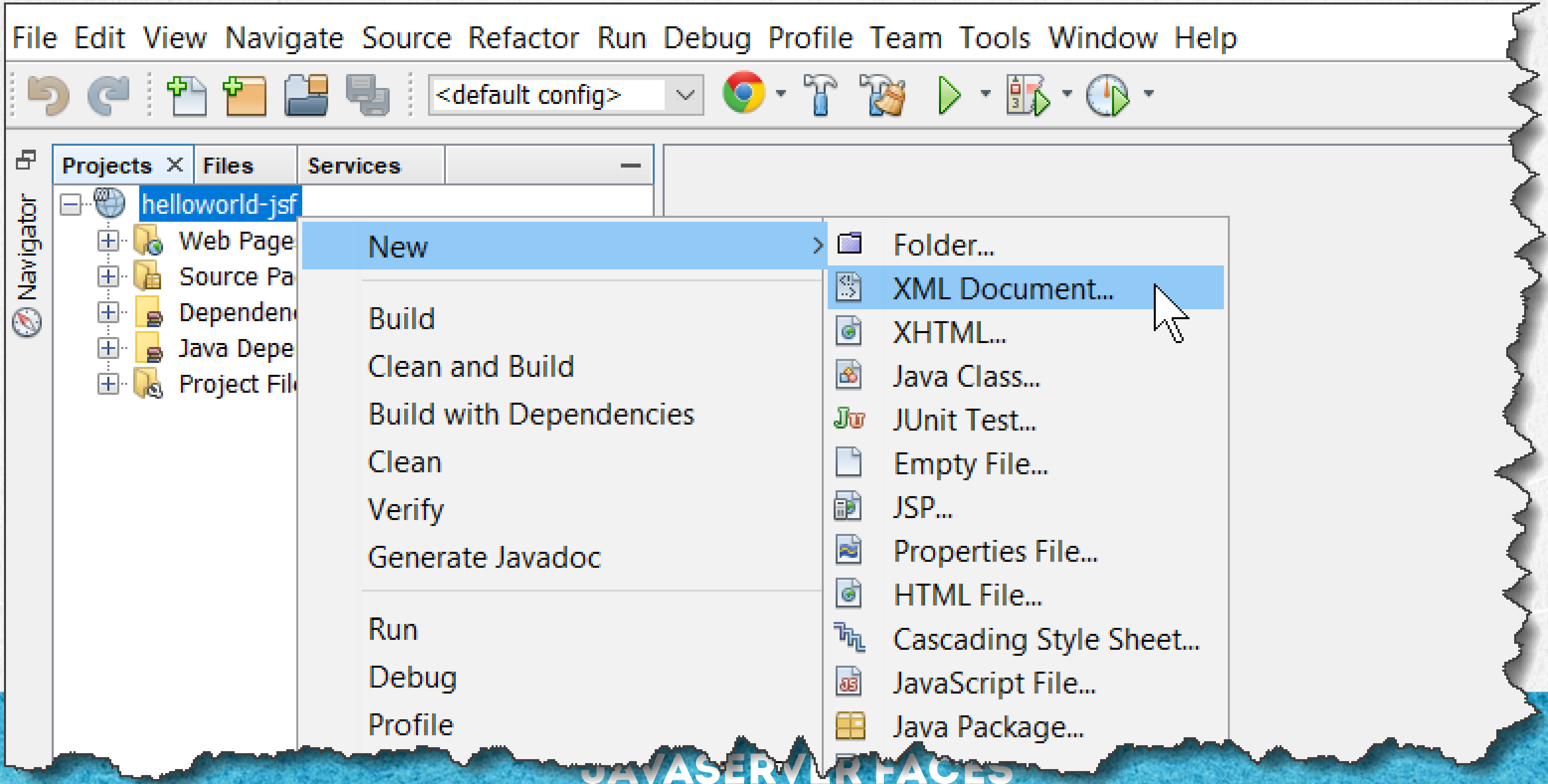
We create a log4j2.xml file. The log4j API allows us to manage the log or log of a Java application in a simpler way.

To use this API it is only necessary to add the libraries of log4j which have already been added since the libraries were added, and the log4j2.xml file somewhere that recognizes the classpath, for example in the src folder of the project.

With this we will be ready to specify what information we want to be sent to the console or other places, such as a file. For more information about this API consult:
https://logging.apache.org/log4j/2.x/

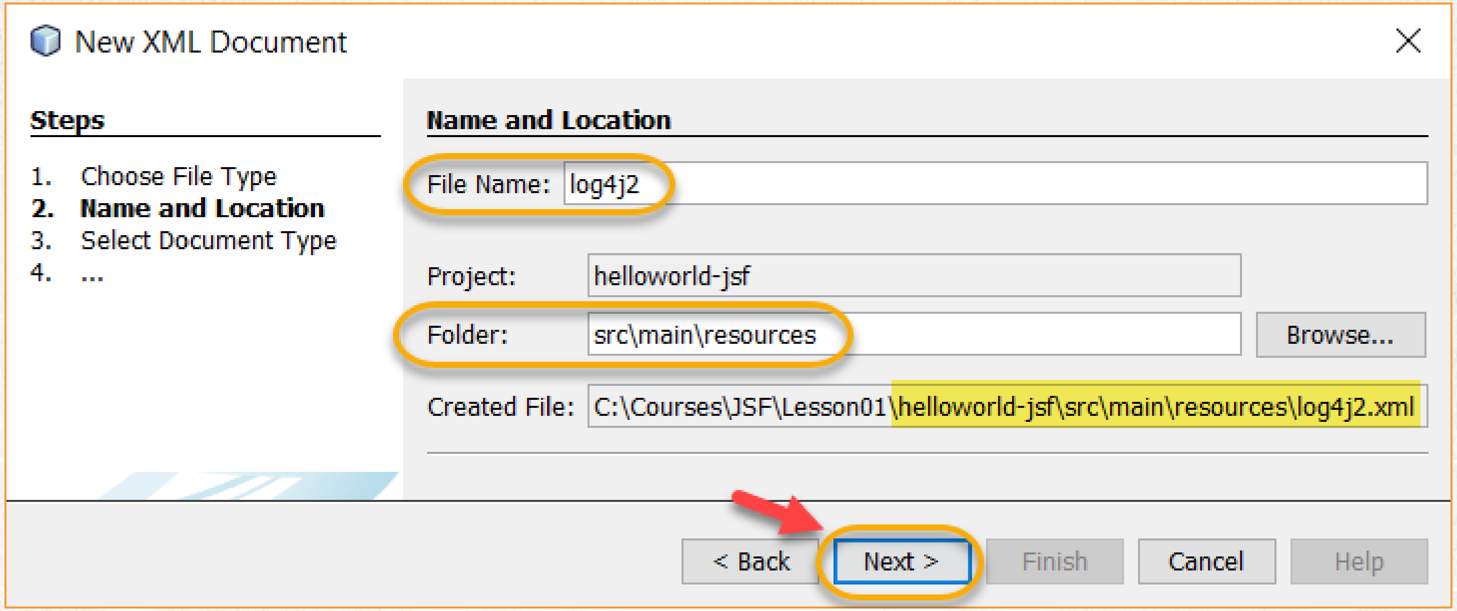
10. CREATE AN XML FILE

- We create the log4j2.xml file:



10. CREATE AN XML FILE

- We create the log4j2.xml file:



New XML Document

Steps

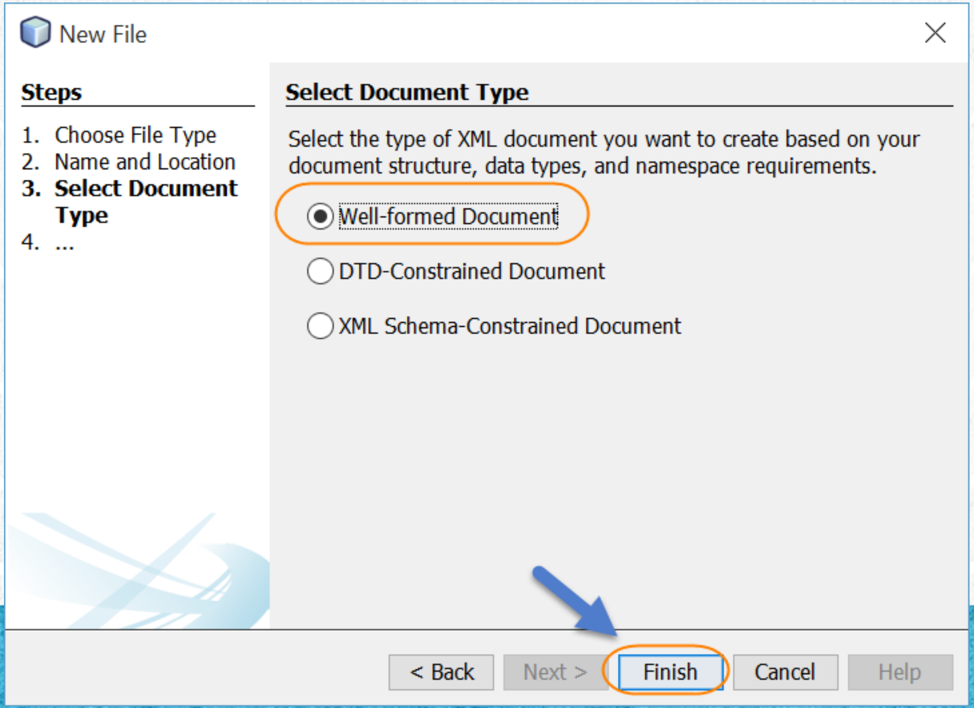
1. Choose File Type
2. **Name and Location**
3. Select Document Type
4. ...

Name and Location

File Name:	log4j2
Project:	helloworld-jsf
Folder:	src\main\resources
Created File:	C:\Courses\JSF\Lesson01\helloworld-jsf\src\main\resources\log4j2.xml

10. CREATE AN XML FILE

- We create the log4j2.xml file. In this step we select any option, it is not important since we are going to overwrite the file:



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

11. MODIFY THE CODE

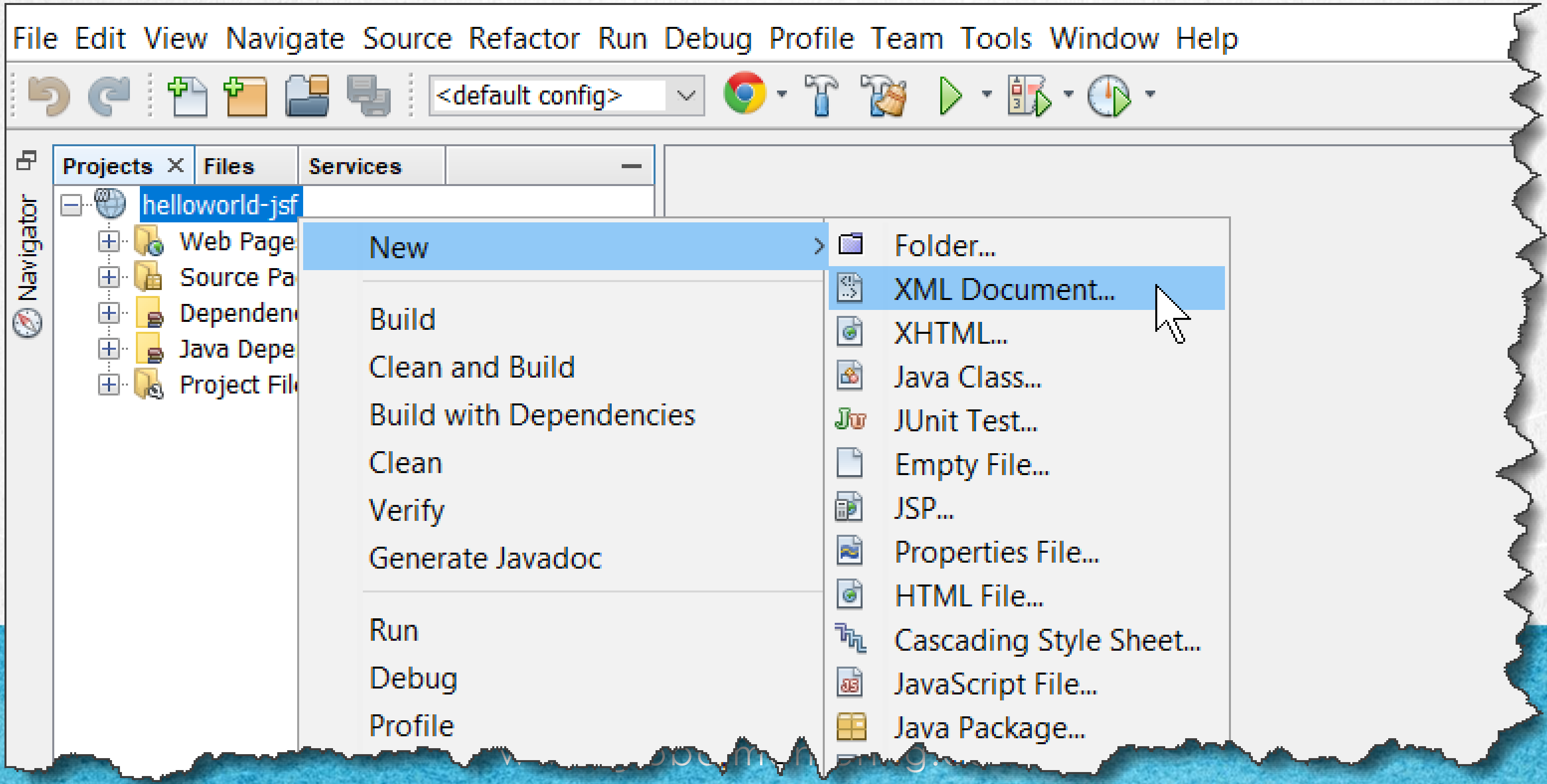
log4j2.xml:

Click to download

```
<?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>
        <Root level="info">
            <AppenderRef ref="Console" />
        </Root>
    </Loggers>
</Configuration>
```

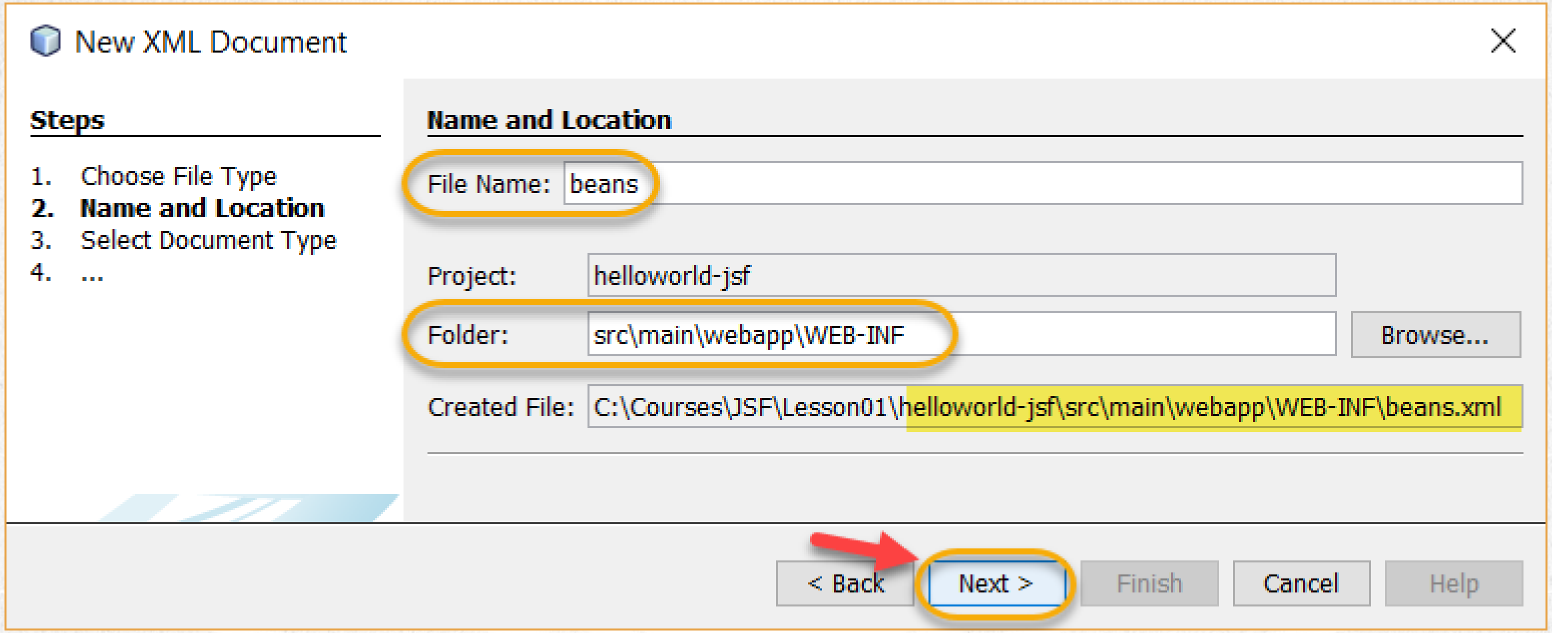
12. CREATE AN XML FILE

- We create the beans.xml file. This file is to activate bean CDI injection, that we will use in the next lessons:



12. CREATE AN XML FILE

- We create the beans.xml file :



New XML Document

Steps

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

Name and Location

File Name: beans

Project: helloworld-jsf

Folder: src\main\webapp\WEB-INF

Created File: C:\Courses\JSF\Lesson01\helloworld-jsf\src\main\webapp\WEB-INF\beans.xml

13. MODIFY THE CODE

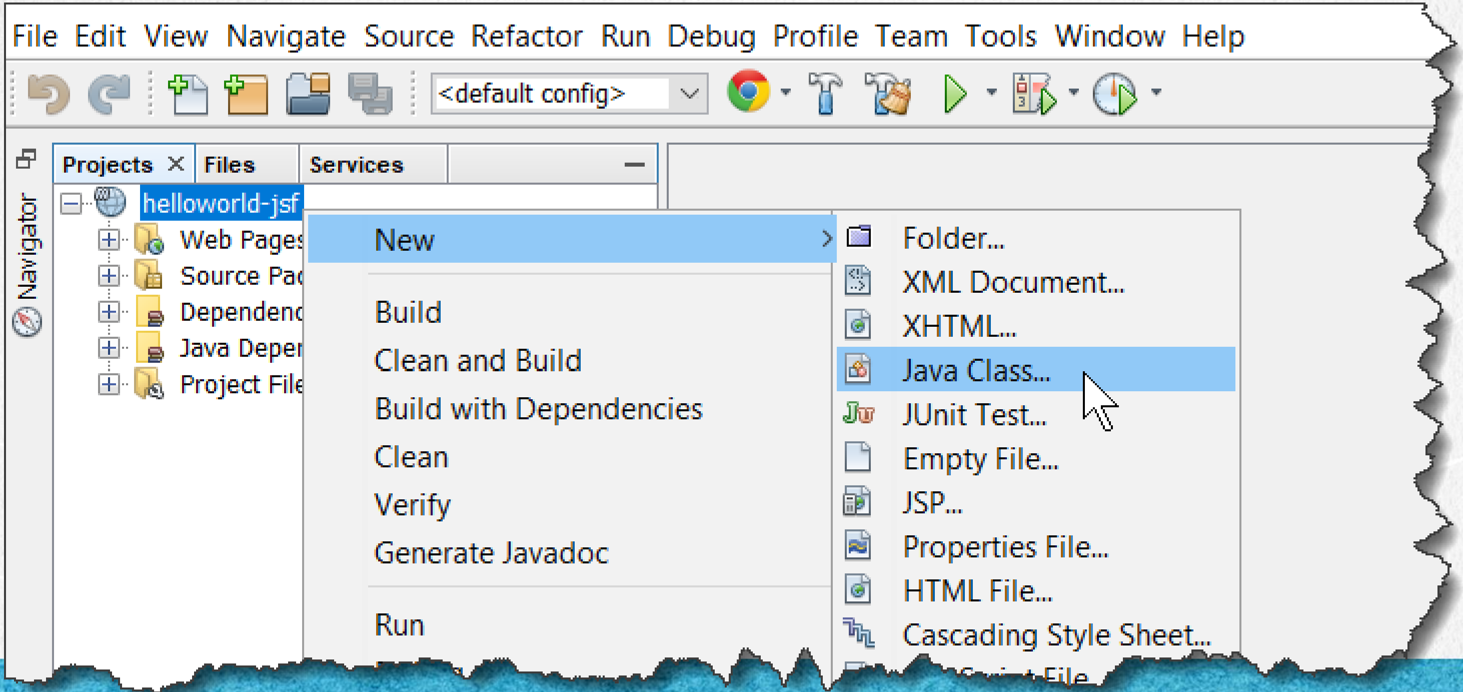
beans.xml:

Click to download

```
<beans 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/beans_2_0.xsd"
  bean-discovery-mode="all"
  version="2.0">

</beans>
```

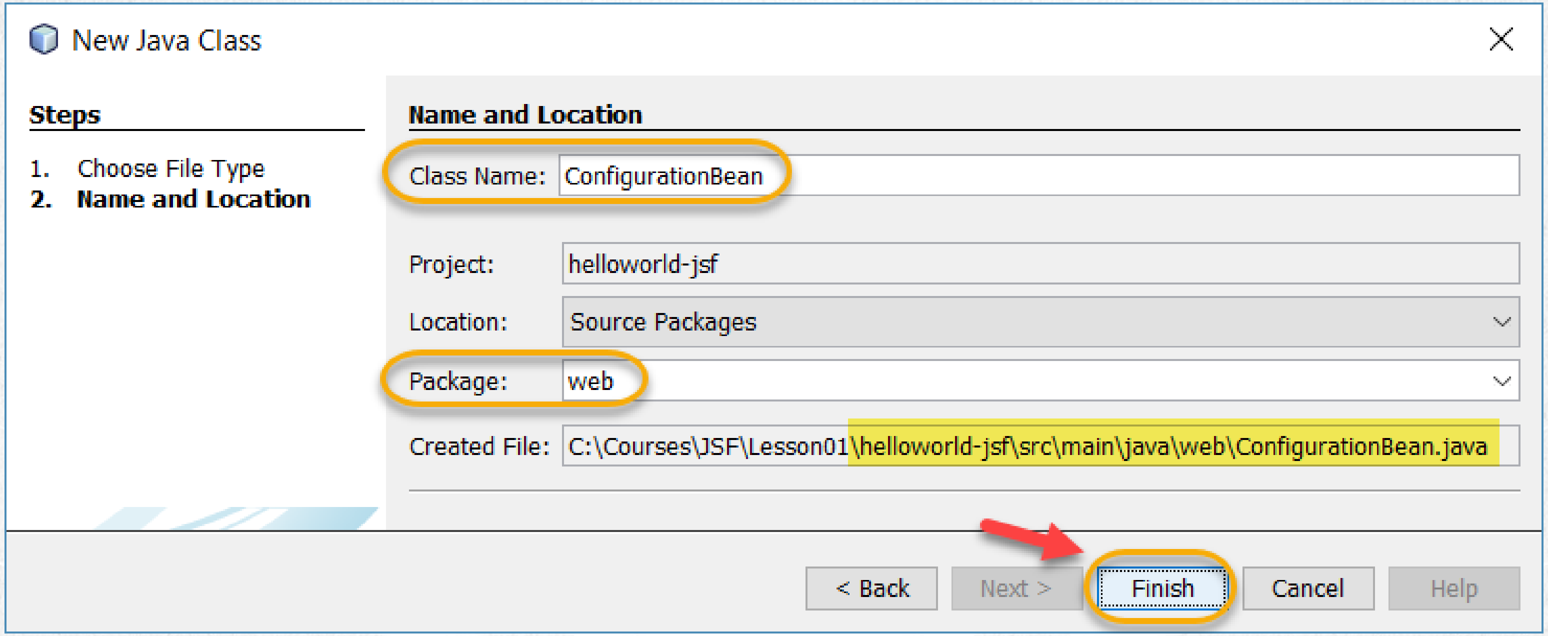
14. CREATE A JAVA FILE

- We create a Java file in order to activate the latest version of JSF.



14. CREATE A JAVA FILE

- We create a Java file in order to activate the latest version of JSF.



Class Name: ConfigurationBean

Project: helloworld-jsf

Location: Source Packages

Package: web

Created File: C:\Courses\JSF\Lesson01\helloworld-jsf\src\main\java\web\ConfigurationBean.java

15. MODIFY THE CODE

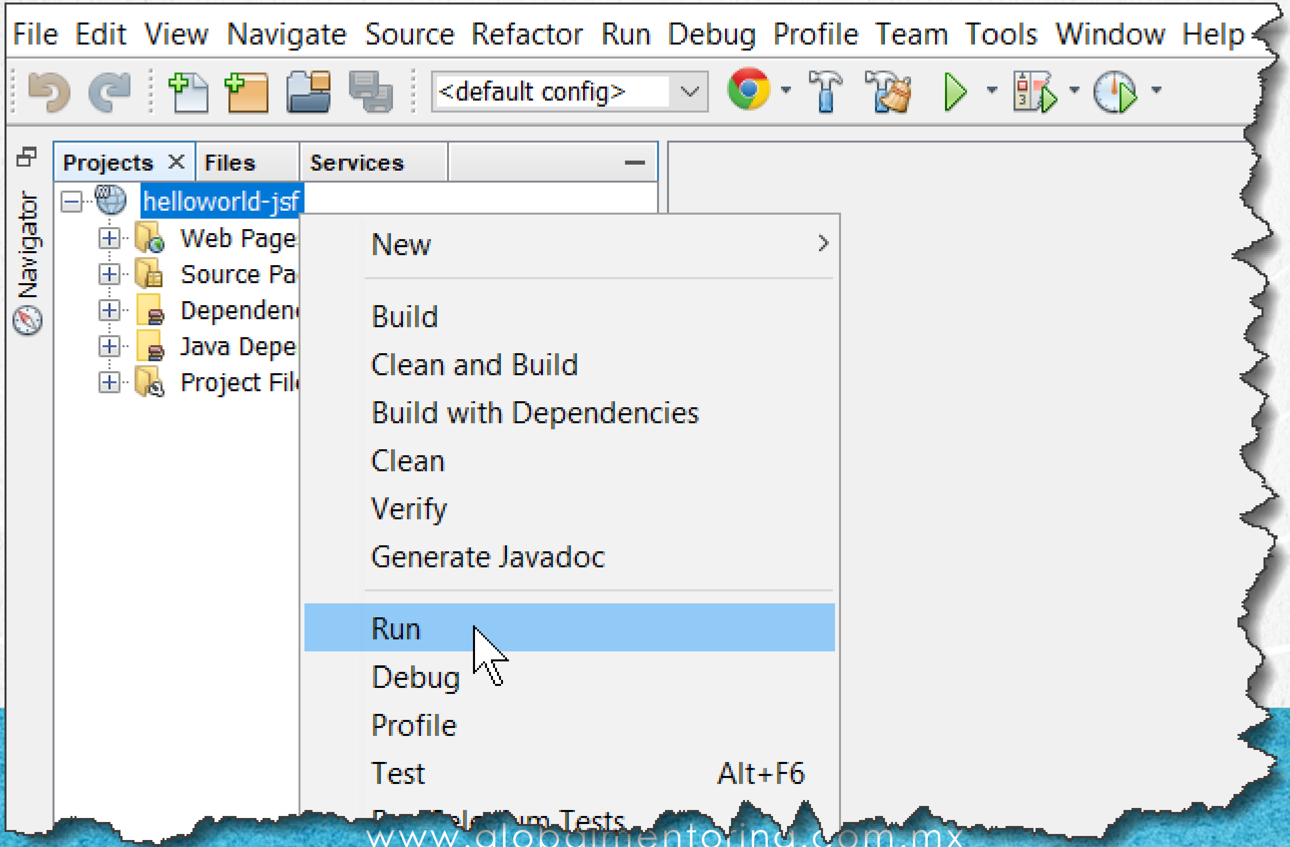
ConfigurationBean.java:

Click to download

```java
package web;

import javax.faces.annotation.FacesConfig;
import static javax.faces.annotation.FacesConfig.Version.JSF_2_3;

@FacesConfig(
// Activates CDI
version = JSF_2_3
)
public class ConfigurationBean {

}
```

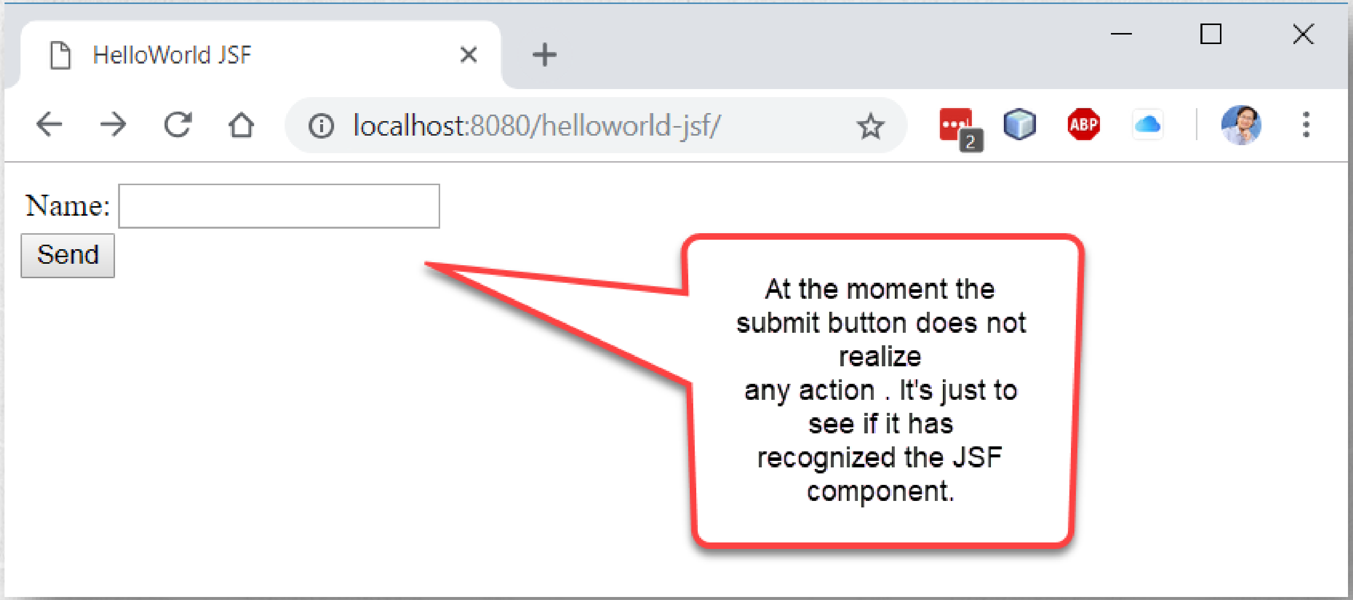
16. EXECUTE THE PROJECT

Execute the project:



16. EXECUTE THE PROJECT

We execute the project obtaining the following result:



At the moment the submit button does not realize any action . It's just to see if it has recognized the JSF component.

EXERCISE CONCLUSION

With this exercise we put into practice the HelloWorld with JSF.

We also saw the creation of the web.xml file, which is one of the files that we will use to make several of the JSF configurations. In particular, we indicate that you recognize the JSF Servlet, and also indicate the start page from this configuration file.

Finally we execute our exercise being able to observe that the components added to the index.xhtml page are displayed correctly.

ONLINE COURSE

JAVASERVER FACES (JSF)

By: Eng. Ubaldo Acosta

JAVASERVER FACES

www.globalmentoring.com.mx