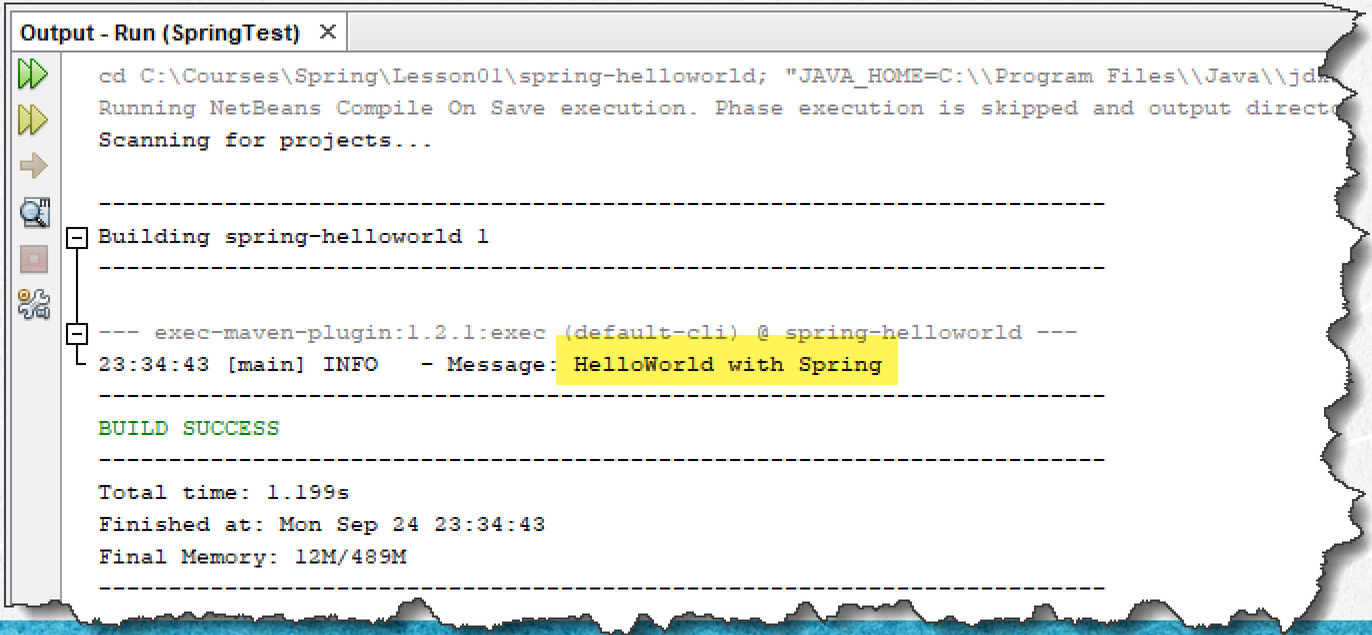
EXERCISE

HELLOWORLD WITH SPRING FRAMEWORK

Global Mentoring

EXERCISE OBJECTIVE

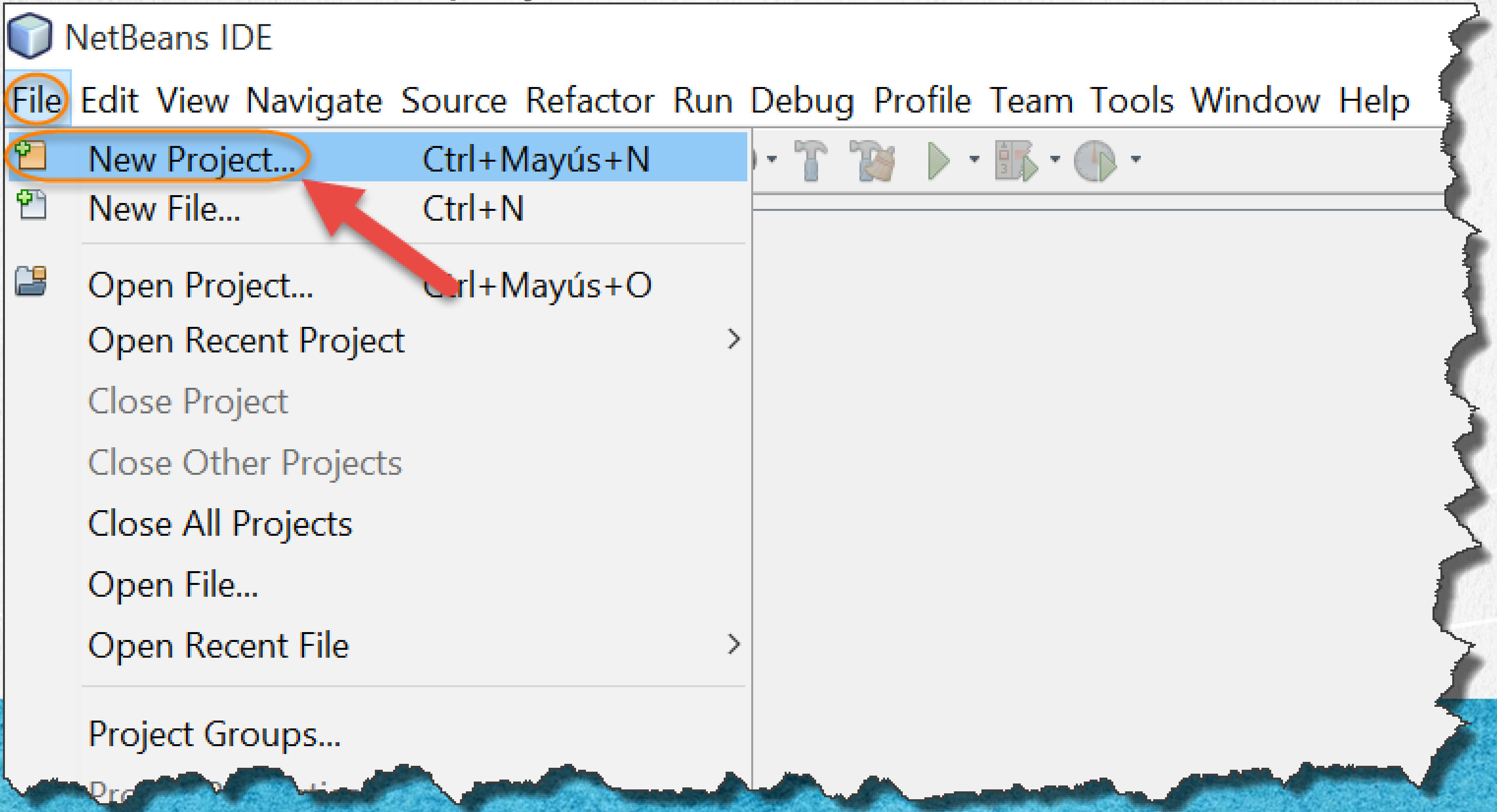
- The objective of the exercise is to set up a HelloWorld project with Spring Framework. We will rely on Maven for the creation of the project. The result should be similar to the following:



```
cd C:\Courses\Spring\Lesson01\spring-helloworld; "JAVA_HOME=C:\\Program Files\\Java\\jd
Running NetBeans Compile On Save execution. Phase execution is skipped and output directo
Scanning for projects...

------------------------------------------------------------------------
Building spring-helloworld 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ spring-helloworld ---
23:34:43 [main] INFO  - Message: HelloWorld with Spring
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 1.199s
Finished at: Mon Sep 24 23:34:43
Final Memory: 12M/489M
------------------------------------------------------------------------
```

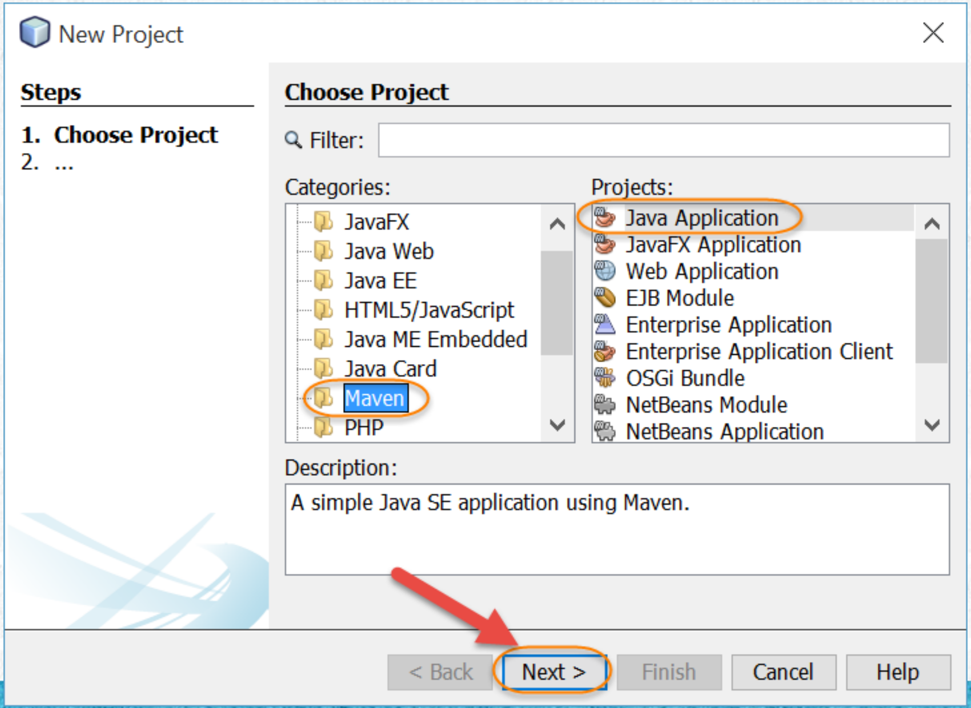
1. CREATE A MAVEN PROJECT

We created a new Maven project:



1. CREATE A MAVEN PROJECT

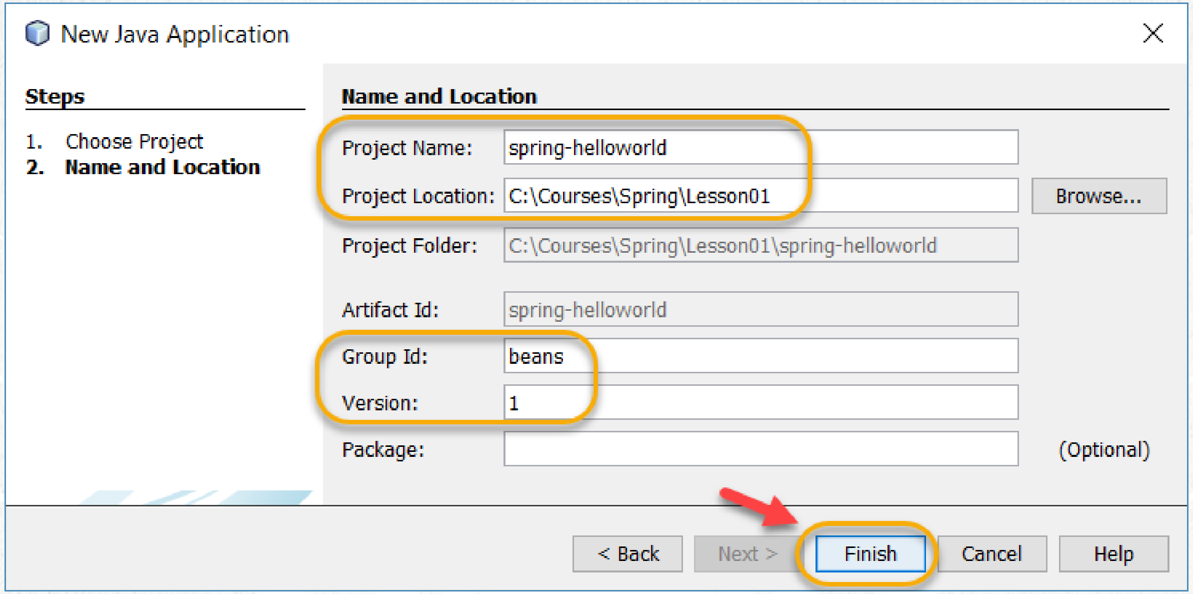
Selecting a new Maven project :



A simple Java SE application using Maven.

1. CREATE A MAVEN PROJECT

We write the following values :



New Java Application

Steps

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

Name and Location

Project Name:	spring-helloworld
Project Location:	C:\Courses\Spring\Lesson01
Project Folder:	C:\Courses\Spring\Lesson01\spring-helloworld
Artifact Id:	spring-helloworld
Group Id:	beans
Version:	1
Package:	(Optional)

2. ADD SPRING LIBRARIES

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

- spring-core
- spring-context
- junit
- log4j

As we progress through the course we will add more features, libraries and configurations.

But at the moment we will use the most basic elements of Spring, and as we advance in the course we will be adding more elements.

2. MODIFY THE FILE

pom.xml:

Click to download

```
<?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>beans</groupId>
    <artifactId>spring-helloworld</artifactId>
    <version>1</version>
    <packaging>jar</packaging>
   <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <spring.version>5.1.0.RELEASE</spring.version>
        <log4j.version>2.11.1</log4j.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
```

2. MODIFY THE FILE

pom.xml:

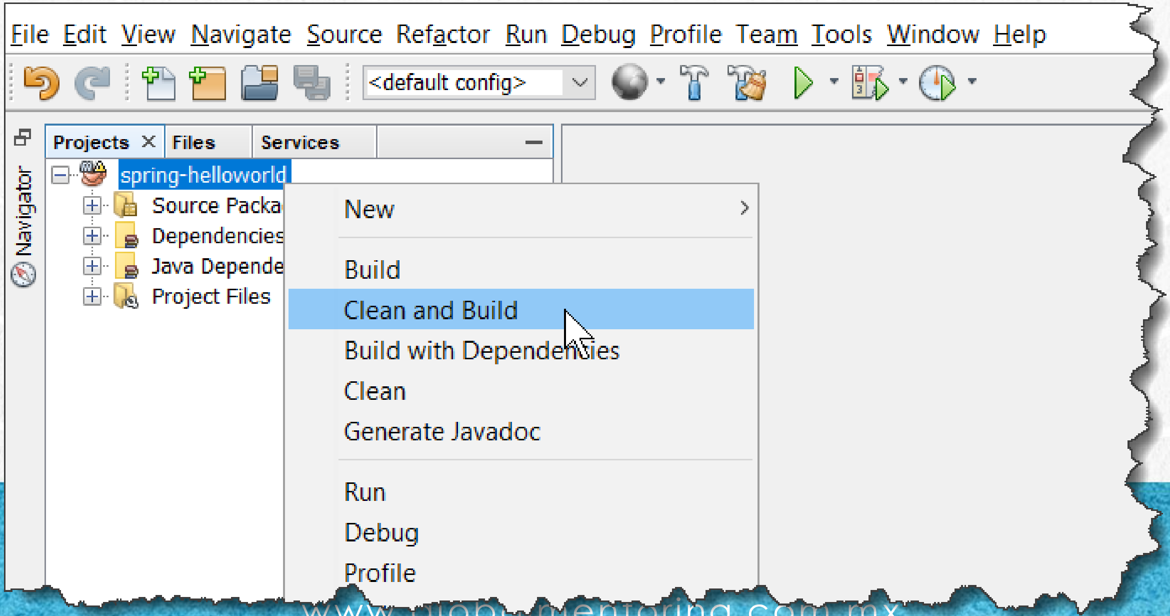
Click to download

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <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>
</project>
```

3. MAVEN REPOSITORY UPDATE)

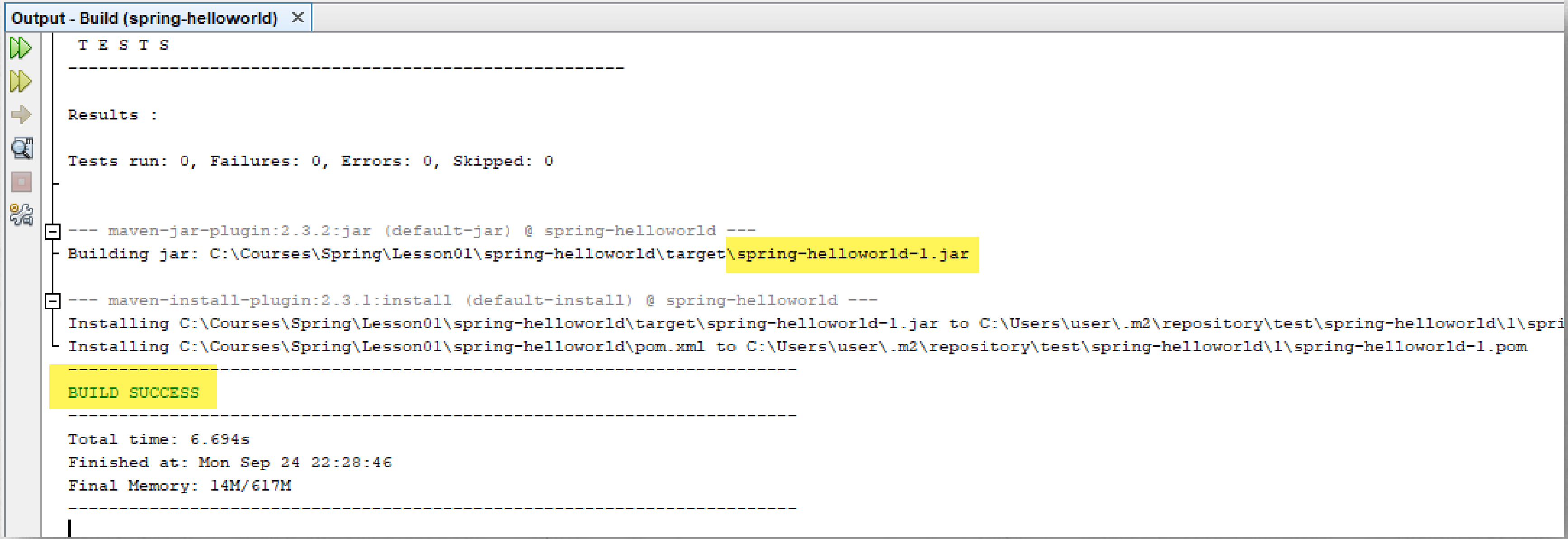
To update the Maven repository, which will be responsible for administering the .jar libraries of our project, it is enough to make clean & build our project

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



3. MAVEN REPOSITORY UPDATE)

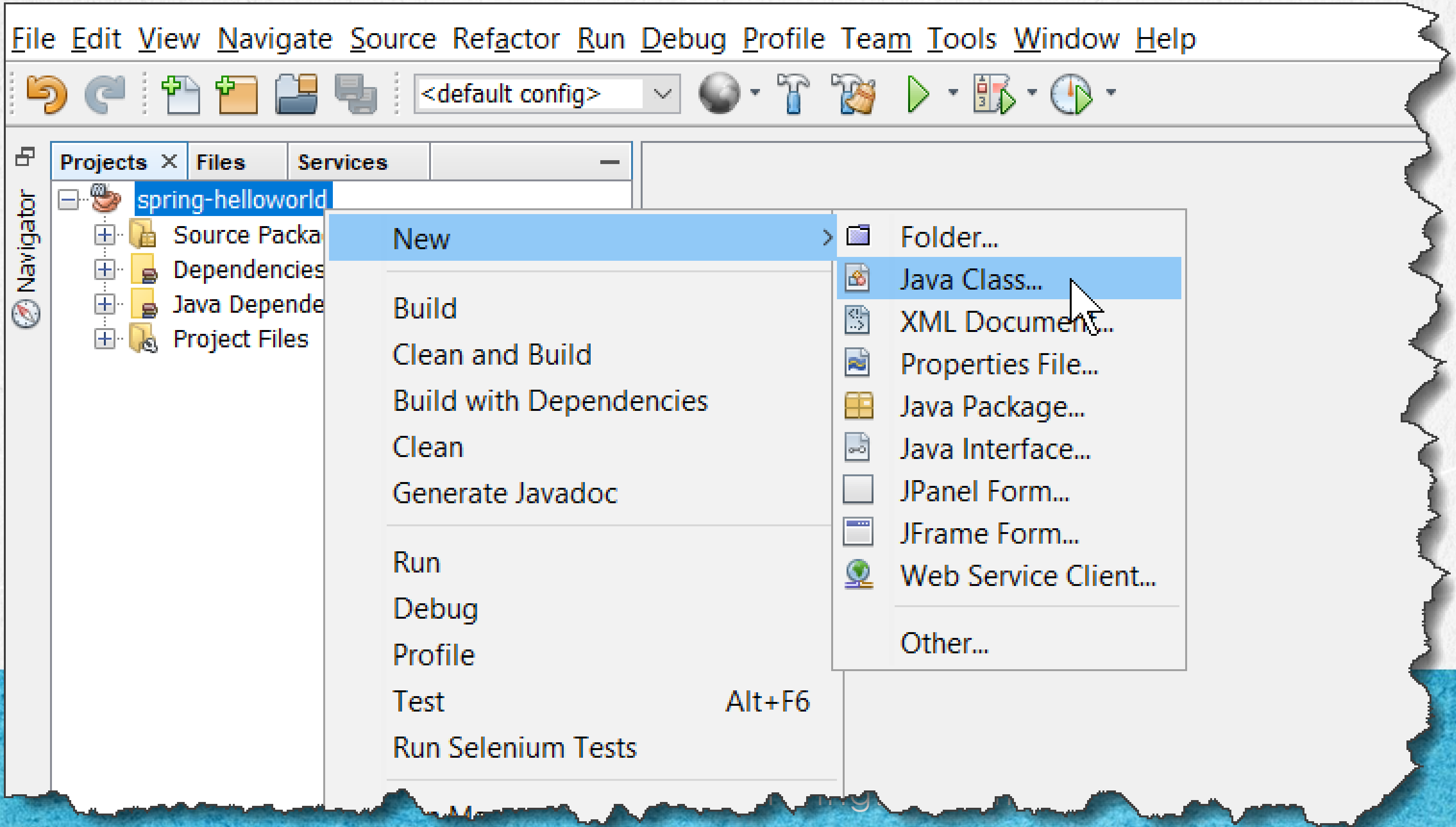
The result is similar to the following:



```
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0


--- maven-jar-plugin:2.3.2:jar (default-jar) @ spring-helloworld ---
Building jar: C:\Courses\Spring\Lesson01\spring-helloworld\target\spring-helloworld-1.jar

--- maven-install-plugin:2.3.1:install (default-install) @ spring-helloworld ---
Installing C:\Courses\Spring\Lesson01\spring-helloworld\target\spring-helloworld-1.jar to C:\Users\user\.m2\repository\test\spring-helloworld\1\spri
Installing C:\Courses\Spring\Lesson01\spring-helloworld\pom.xml to C:\Users\user\.m2\repository\test\spring-helloworld\1\spring-helloworld-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 6.694s
Finished at: Mon Sep 24 22:28:46
Final Memory: 14M/617M
------------------------------------------------------------------------
```

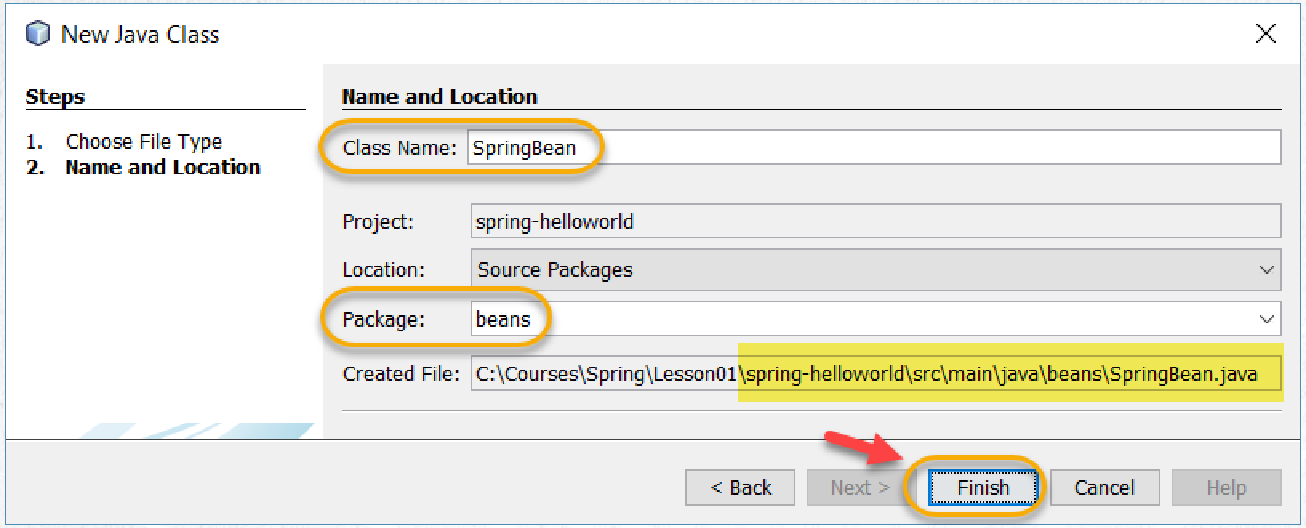
4. Create a Java Class

Next we create the class SpringBean.java:



4. CREATE A JAVA CLASS

Next we create the class SpringBean.java:



New Java Class

Steps

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

Name and Location

Class Name: SpringBean

Project: spring-helloworld

Location: Source Packages

Package: beans

Created File: C:\Courses\Spring\Lesson01\spring-helloworld\src\main\java\beans\SpringBean.java

5. MODIFY THE FILE

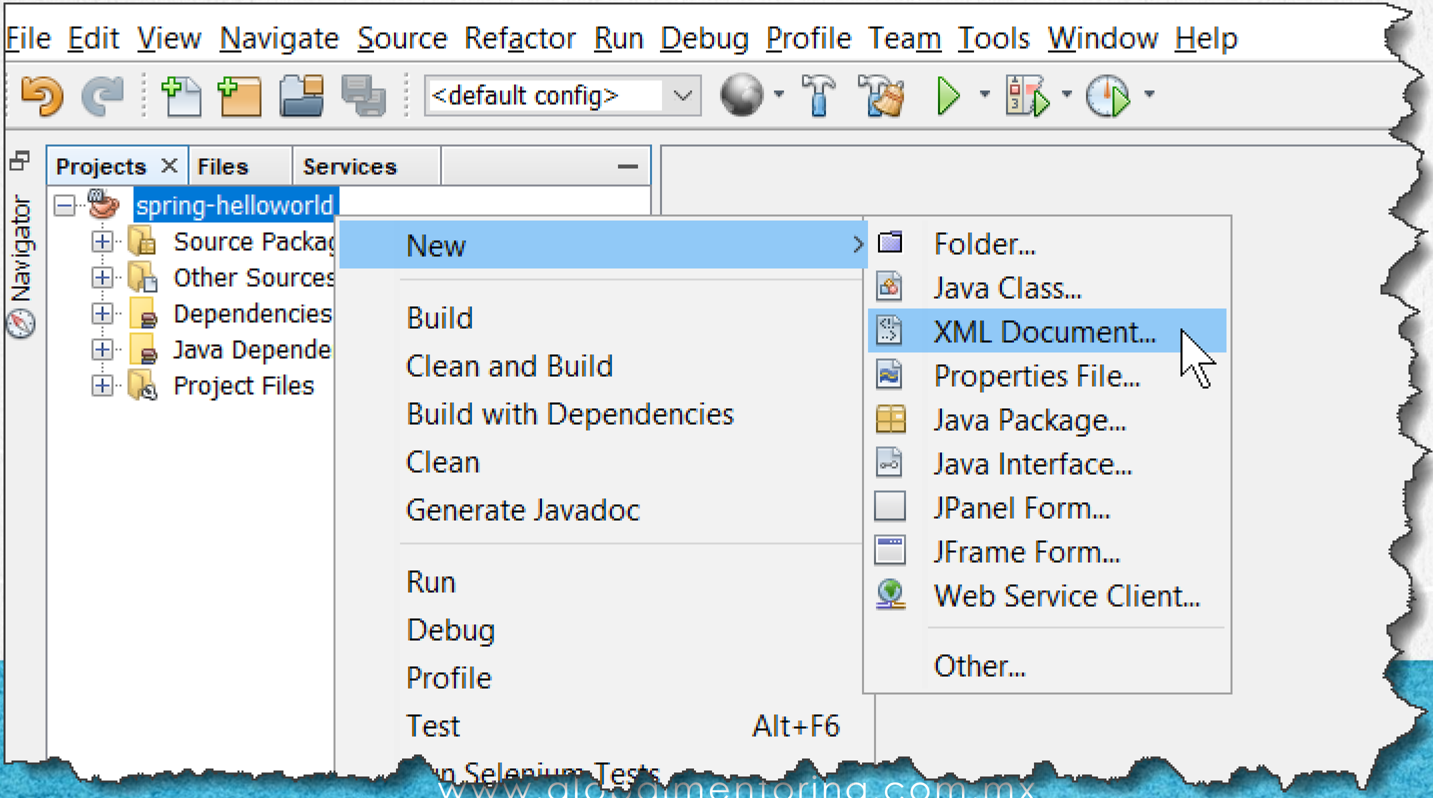
SpringBean.java:

Click to download

```
package beans;

public class SpringBean {

    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

}
```

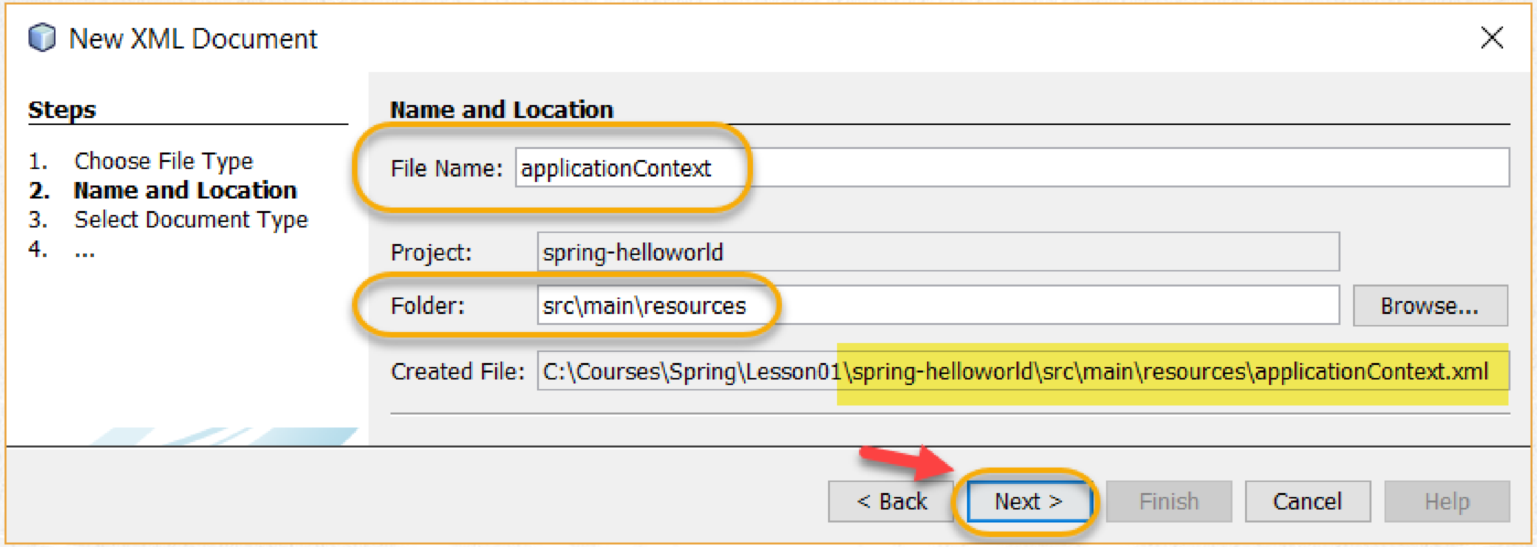
6. CREATE THE APPLICATIONCONTEXT.XML FILE

Next we are going to create the applicationContext.xml file



6. CREATE THE APPLICATIONCONTEXT.XML FILE

Next we are going to create the applicationContext.xml file. We create this file in the folder shown:



New XML Document

Steps

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

Name and Location

File Name: applicationContext

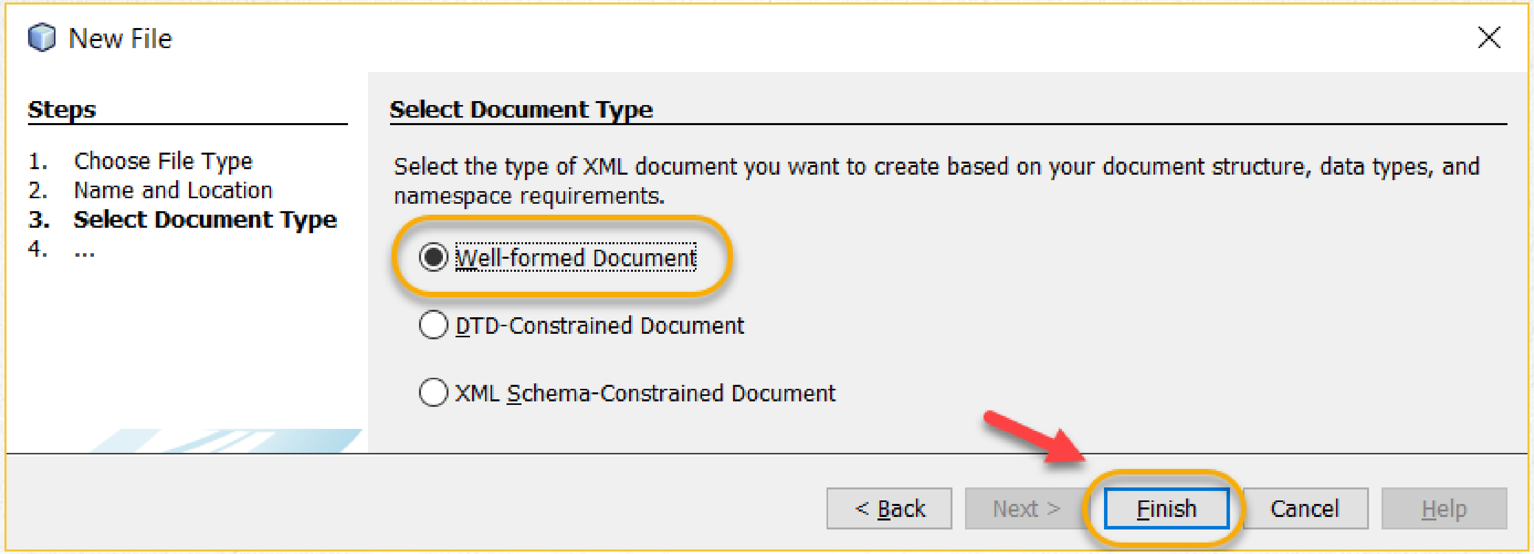
Project: spring-helloworld

Folder: src\main\resources

Created File: C:\Courses\Spring\Lesson01\spring-helloworld\src\main\resources\applicationContext.xml

6. CREATE THE APPLICATIONCONTEXT.XML FILE

Select the option shown below:


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 APPLICATIONCONTEXT.XML FILE

We add the SpringBean definition to be able to use it between the tag <beans>

```
<bean id= "myBean" class="beans.SpringBean">
   <property name= "menssage" value="HelloWorld with Spring" />
</bean>
```

The result should be similar to the one shown below:

7. MODIFY THE FILE

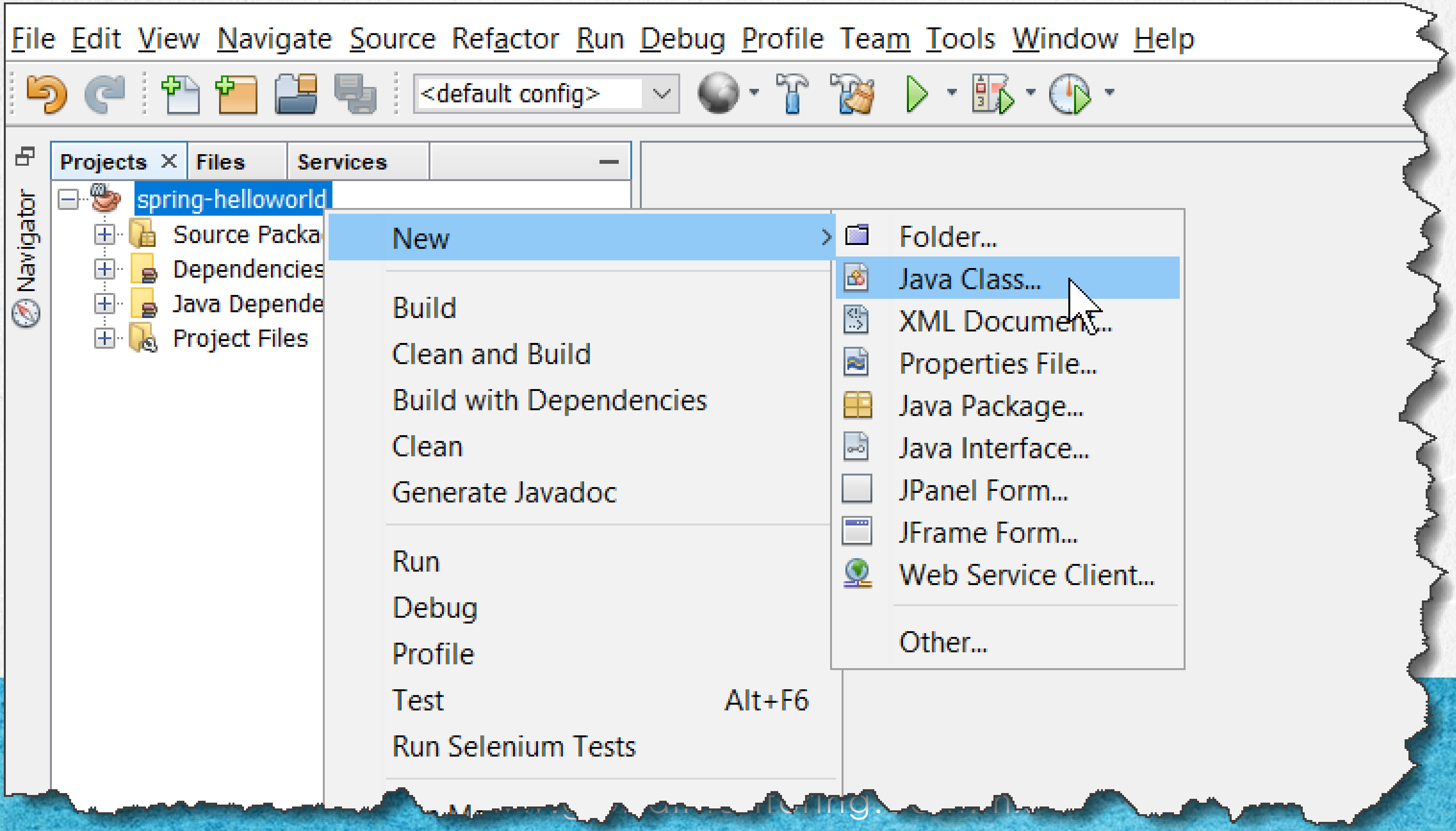
applicationContext.xml

Click to download

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
       xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation = "http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="myBean" class="beans.SpringBean">
        <property name="message" value="HelloWorld with Spring" />
    </bean>
</beans>
```

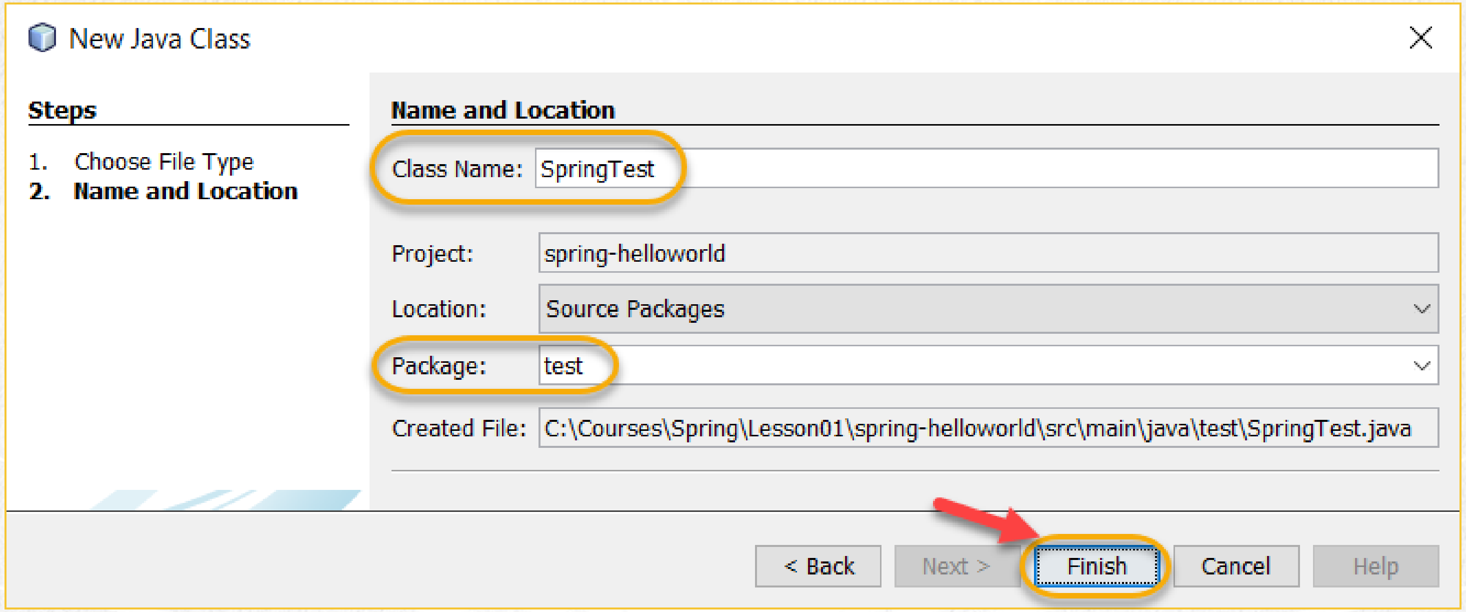
8. CREATE A TEST CLASS

We created a TestSpring.java class to test the Spring configuration:



8. CREATE A TEST CLASS

We created a SpringTest.java class to test the Spring configuration:



New Java Class

Steps

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

Name and Location

Class Name: SpringTest

Project: spring-helloworld

Location: Source Packages

Package: test

Created File: C:\Courses\Spring\Lesson01\spring-helloworld\src\main\java\test\SpringTest.java

9. MODIFY THE CODE

SpringTest.java:

Click to download

```
package test;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.apache.logging.log4j.*;
import beans.SpringBean;

public class SpringTest {

    public static void main(String[] args) {
        Logger log = LogManager.getRootLogger();
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BeanFactory factory = context;
        SpringBean myBean = (SpringBean) factory.getBean("myBean");

        log.info("Message: " + myBean.getMessage());
    }
}
```

10. CREATE AN XML FILE

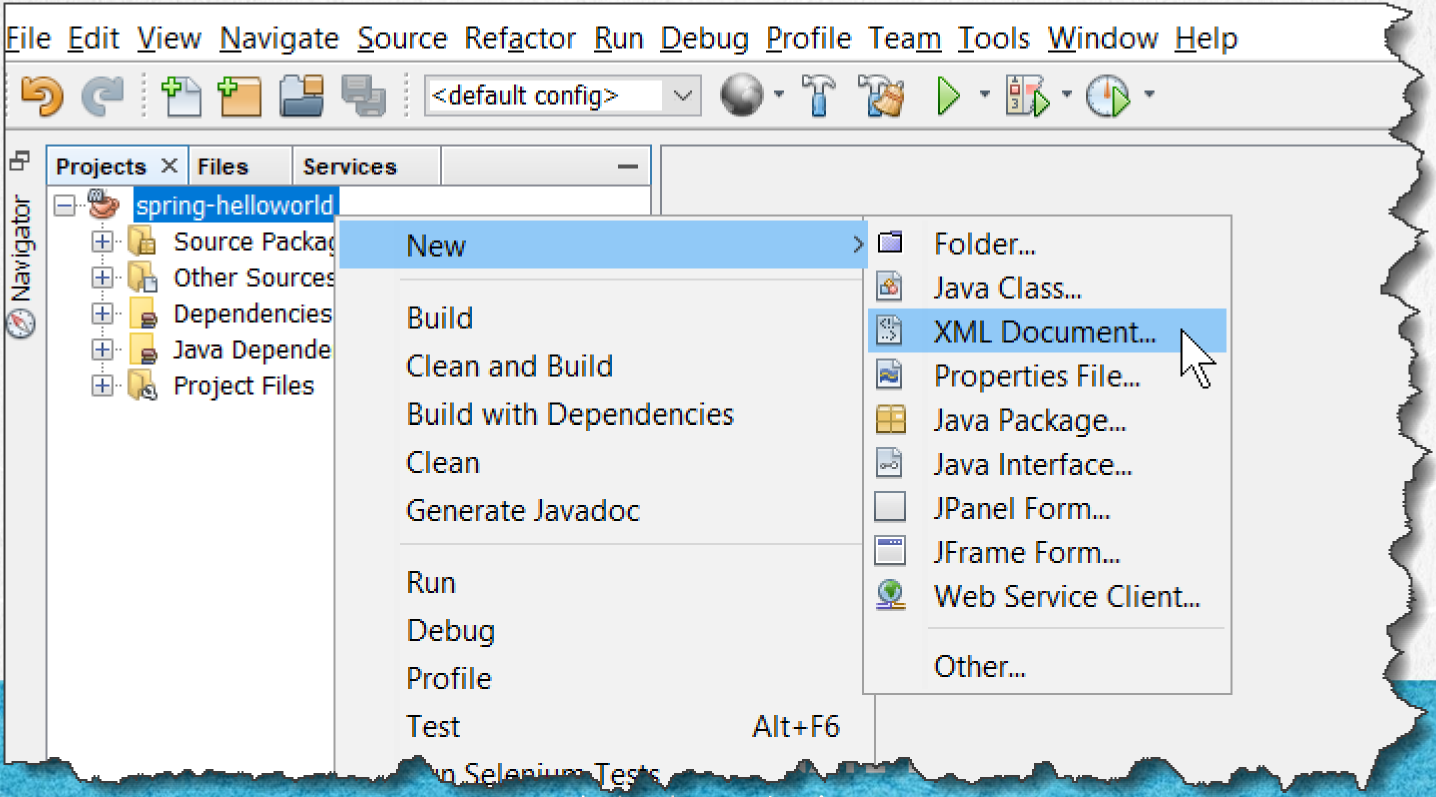
We create a log4j2.xml file. The log4j API allows us to manage the log or log of a Java application in a simpler way.

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/

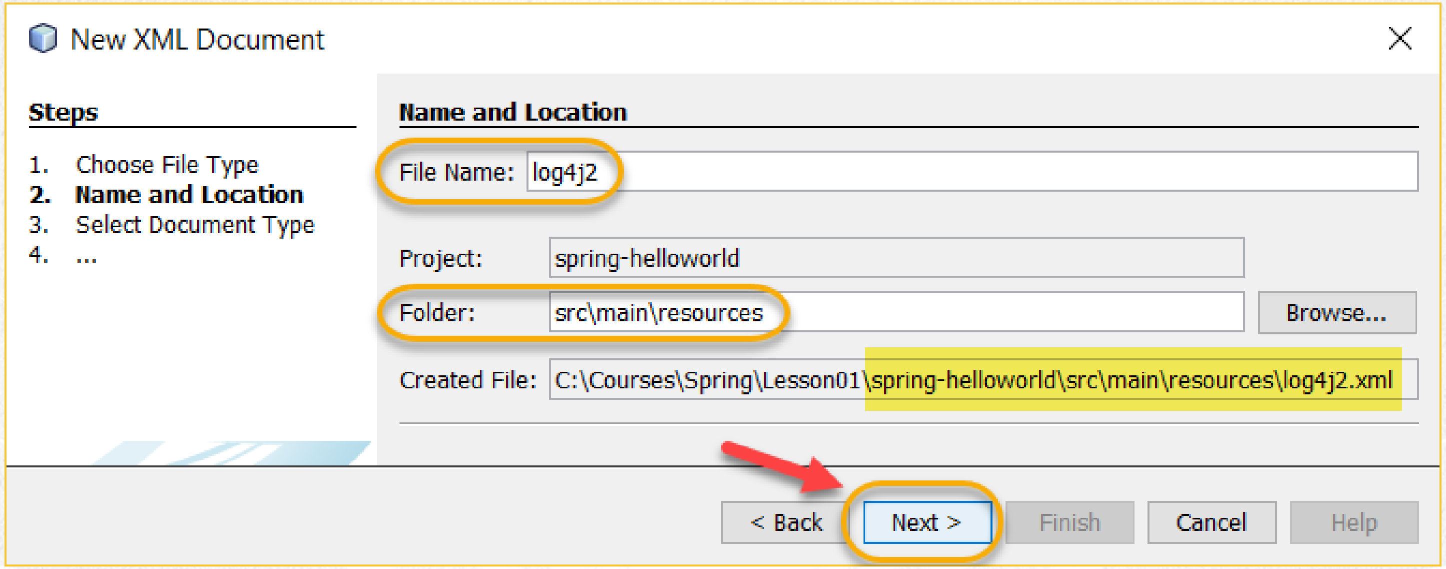
10. CREATE AN XML FILE

- Create the log4j2.xml file:



10. CREATE AN XML FILE

- 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

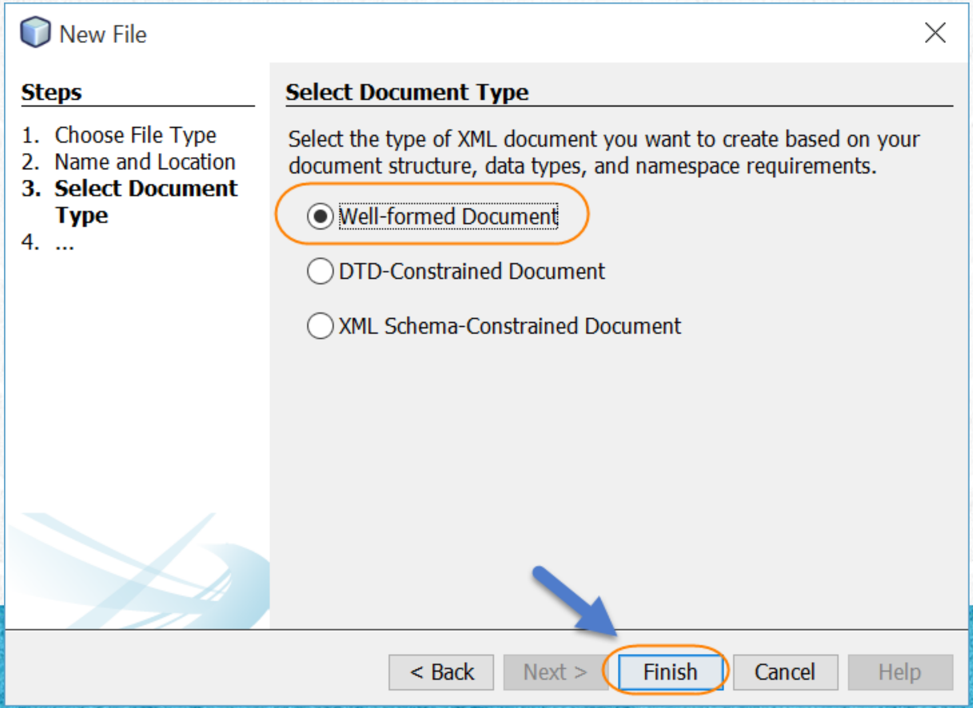
Project: spring-helloworld

Folder: src\main\resources

Created File: C:\Courses\Spring\Lesson01\spring-helloworld\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 FILE

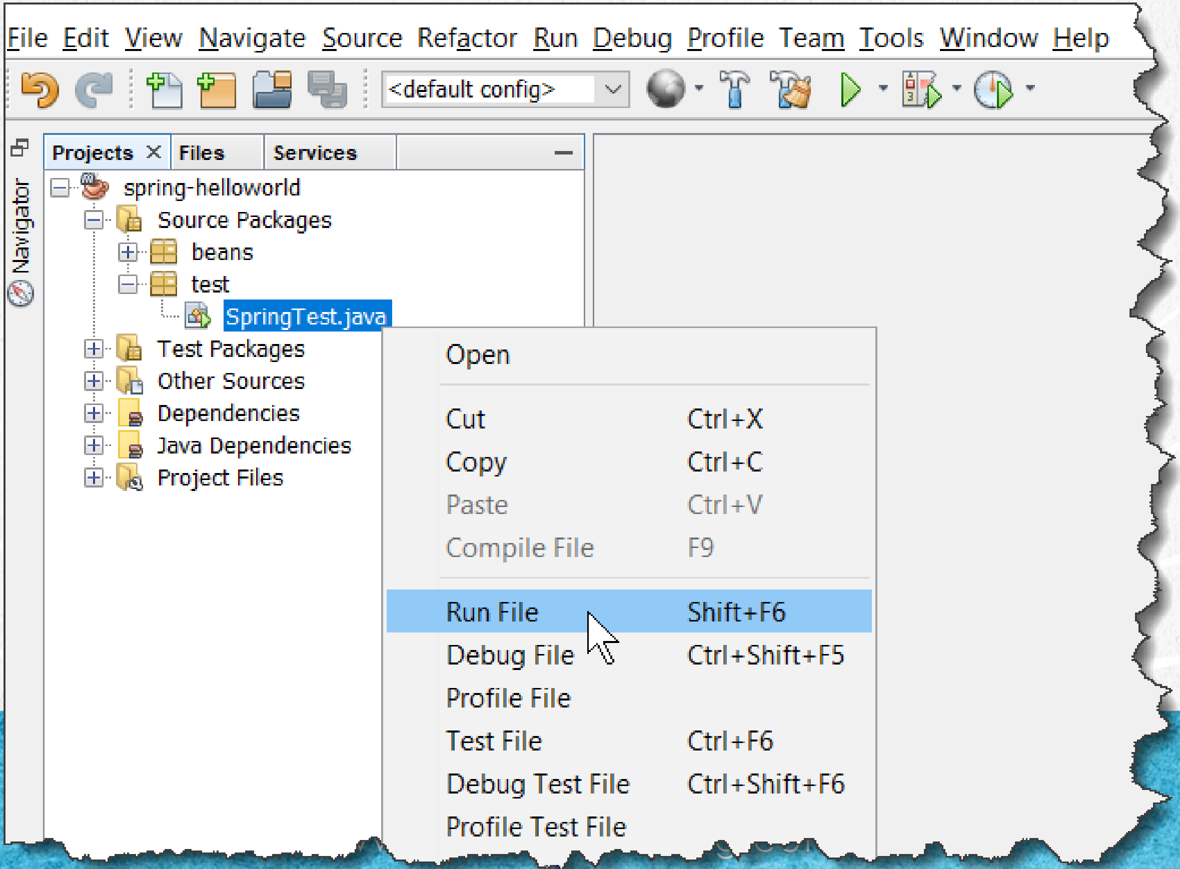
log4j2.xml:

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

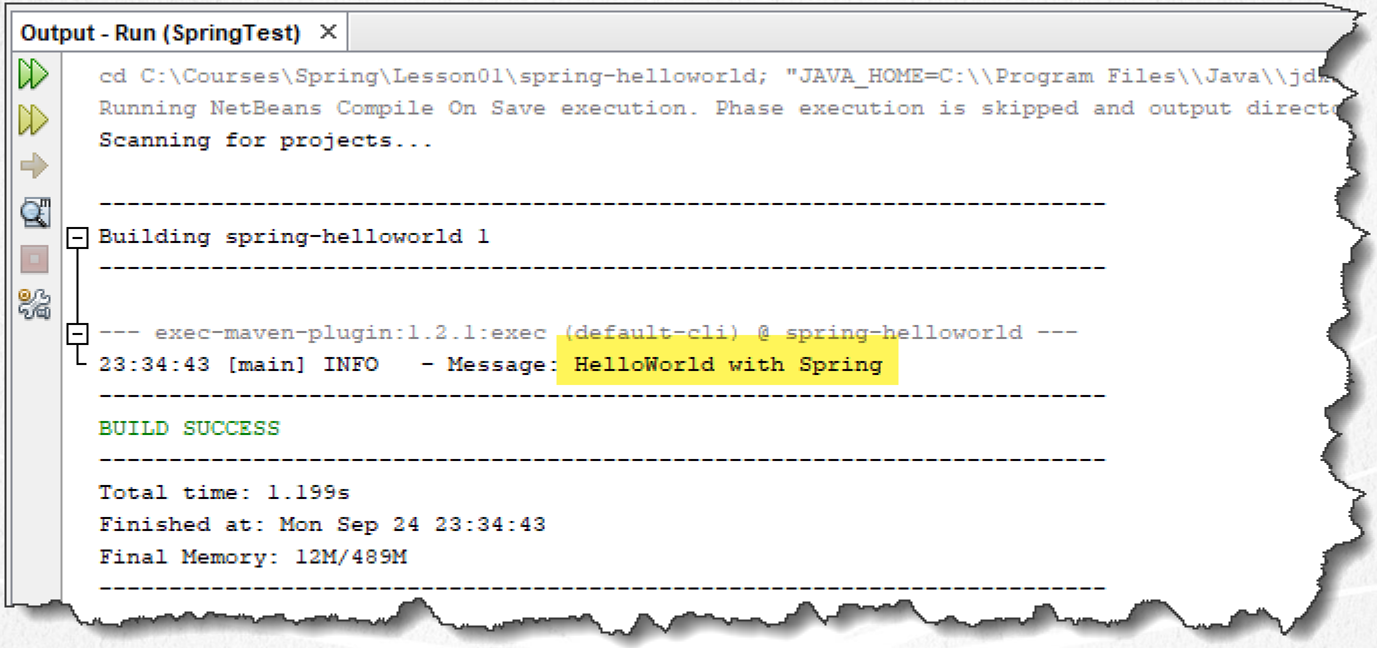
12. EXECUTE THE PROJECT

Execute the Project:



12. EXECUTE THE PROJECT

We execute the project obtaining the following result:



```
cd C:\Courses\Spring\Lesson01\spring-helloworld; "JAVA_HOME=C:\\Program Files\\Java\\jdk
Running NetBeans Compile On Save execution. Phase execution is skipped and output directo
Scanning for projects...

------------------------------------------------------------------------
Building spring-helloworld 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ spring-helloworld ---
23:34:43 [main] INFO   - Message: HelloWorld with Spring
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 1.199s
Finished at: Mon Sep 24 23:34:43
Final Memory: 12M/489M
------------------------------------------------------------------------
```

EXERCISE CONCLUSION

With this exercise we put into practice the HelloWorld with Spring Framework. We use the latest versions of the framework to configure it.

We also saw the creation of the applicationContext.xml file, which will perform all the "magic" of the Spring framework, although it is worth mentioning that each time it needs fewer configurations to perform the Spring configuration, since the use of annotations has almost replaced complete the need for the xml configuration in the applicationContext.xml file

In this exercise we created a JavaBean called SpringBean, we added the configuration of the injection of values via the applicationContext.xml file, and finally we did a test to see if the configuration we made was the correct one. With this we can continue with our Spring Framework Course.

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta