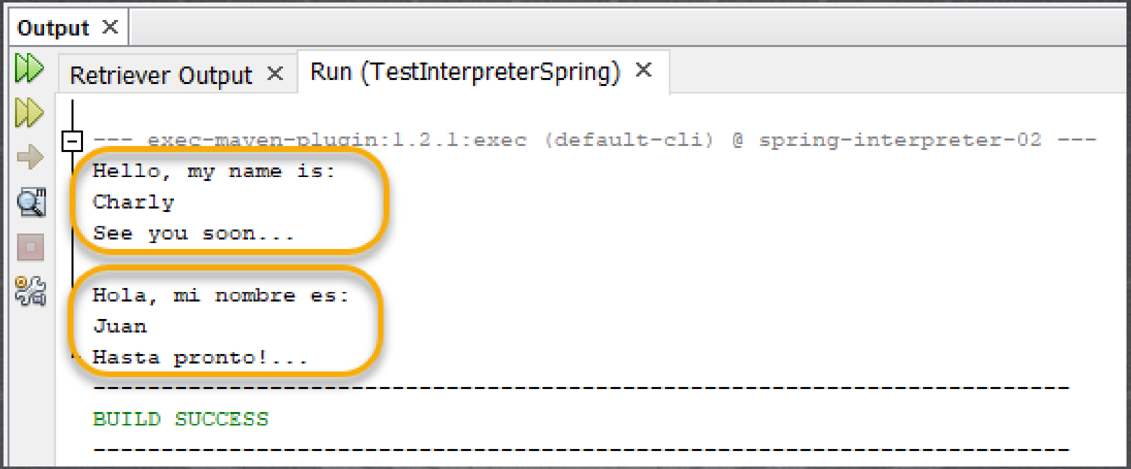
EXERCISE

PROJECT OF INTERPRETER V2 WITH SPRING FRAMEWORK

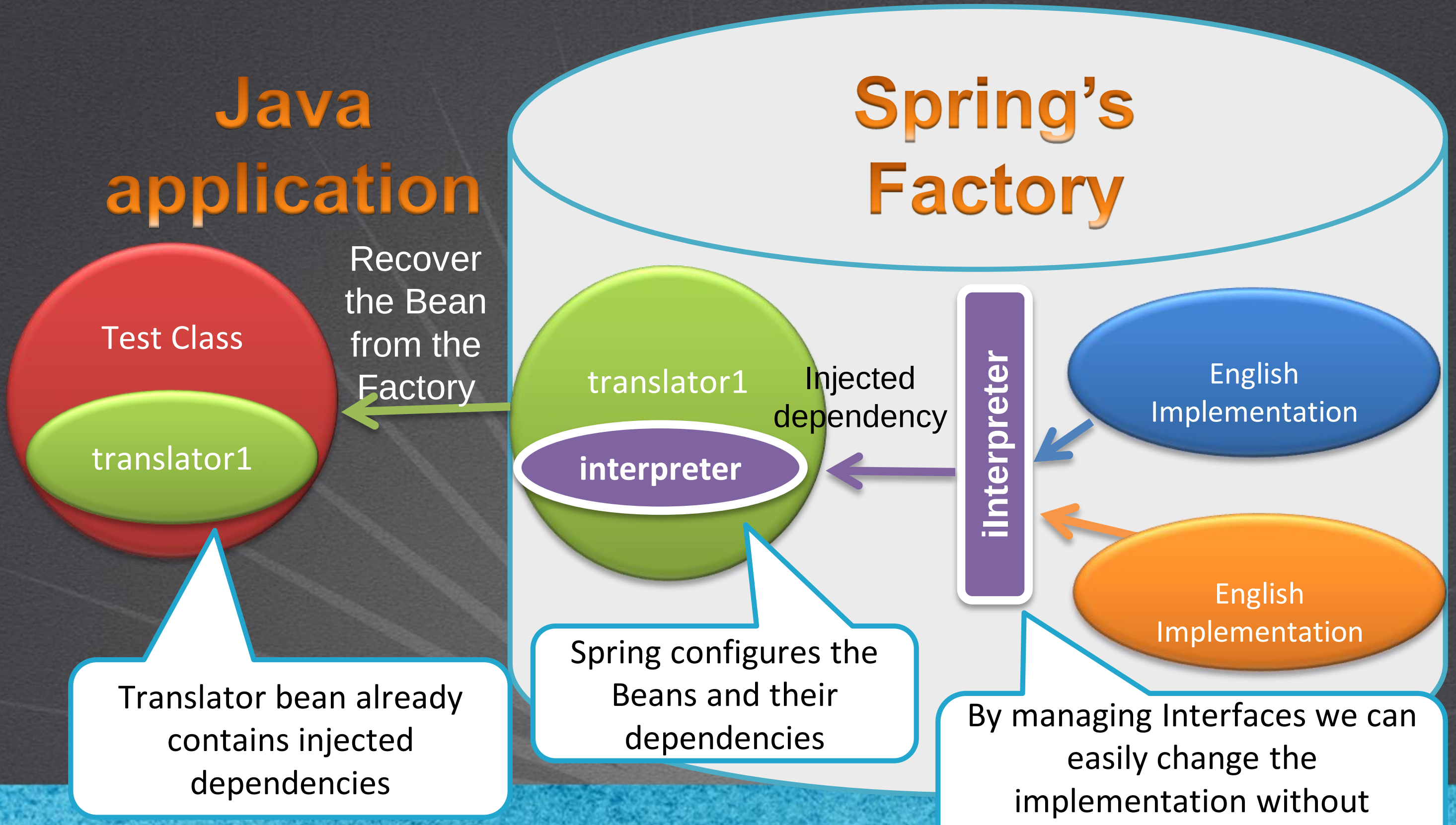
EXERCISE OBJECTIVE

Create the Interpreter project with the Spring Framework. The final result is as follows:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ spring-interpreter-02 ---
Hello, my name is:
Charly
See you soon...

Hola, mi nombre es:
Juan
Hasta pronto!...
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

INTERPRETER VERSION 2



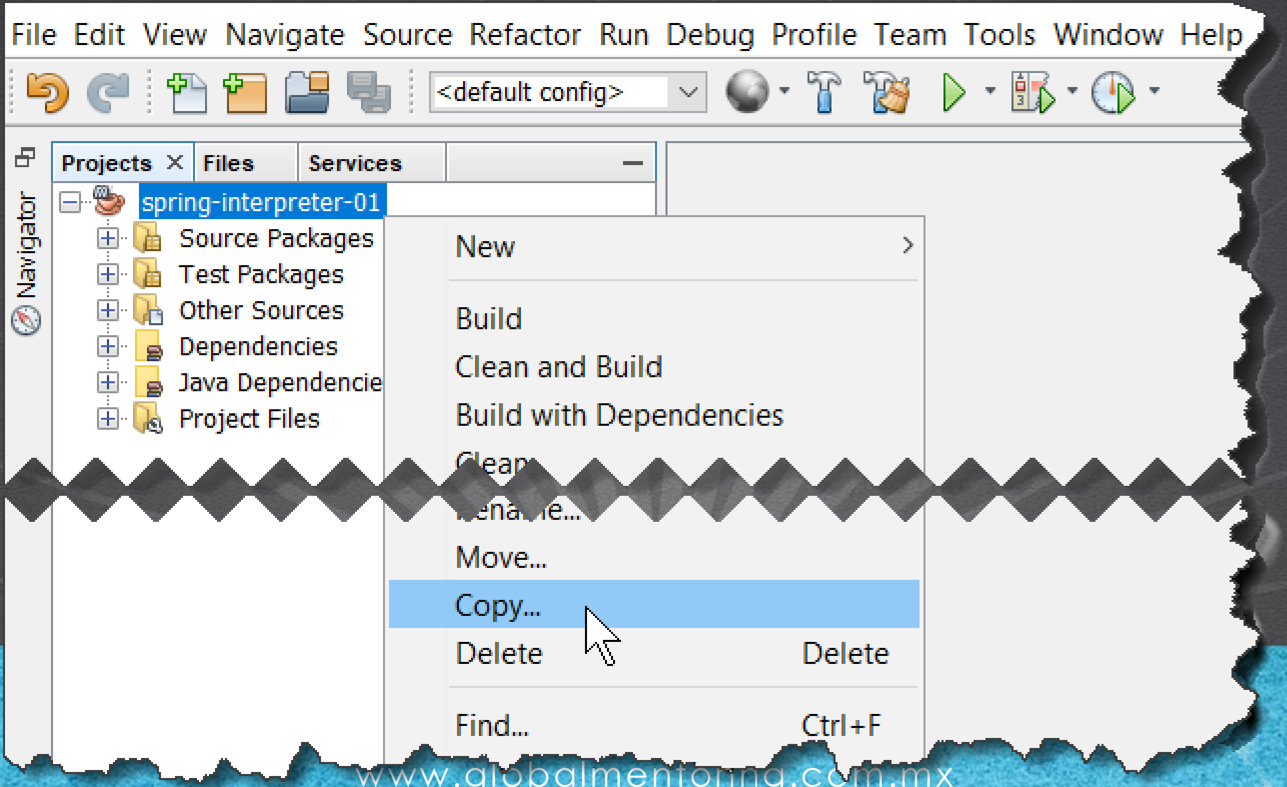
Translator bean already contains injected dependencies

Spring configures the Beans and their dependencies

By managing Interfaces we can easily change the implementation without affecting the translator's code

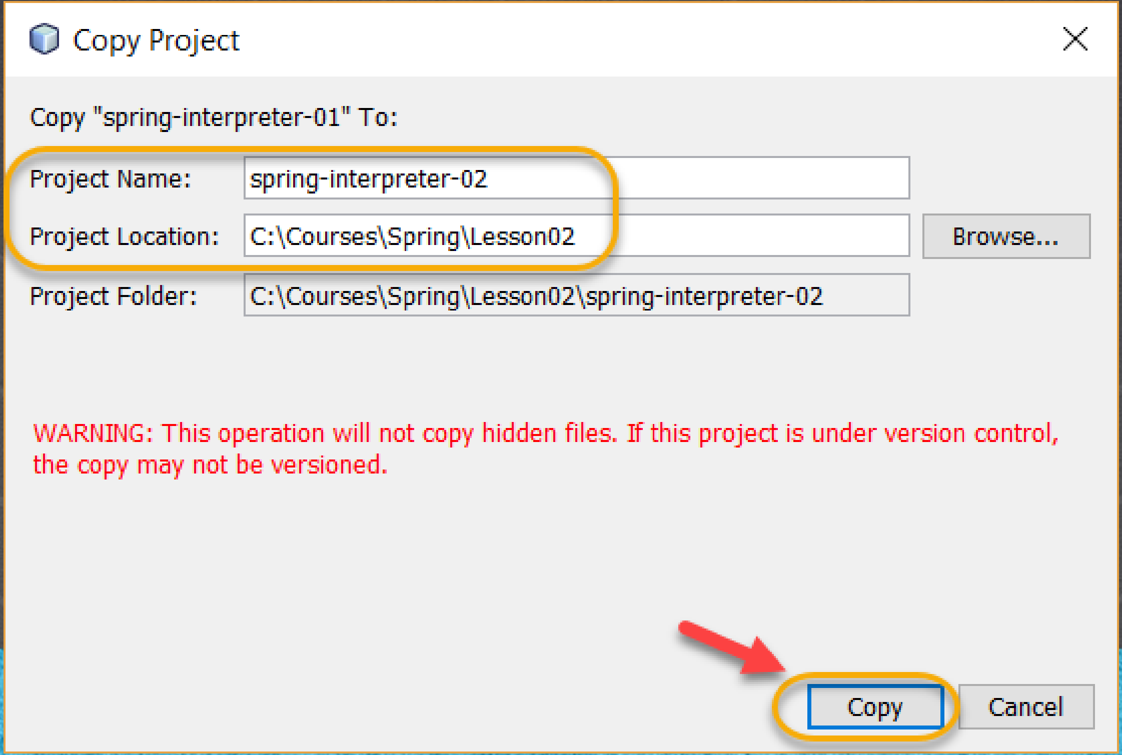
1. COPY THE PROJECT

We copy the previous project to generate version 2:



1. COPY THE PROJECT

We copy the previous project to generate version 2:



Copy "spring-interpreter-01" To:

Project Name: spring-interpreter-02

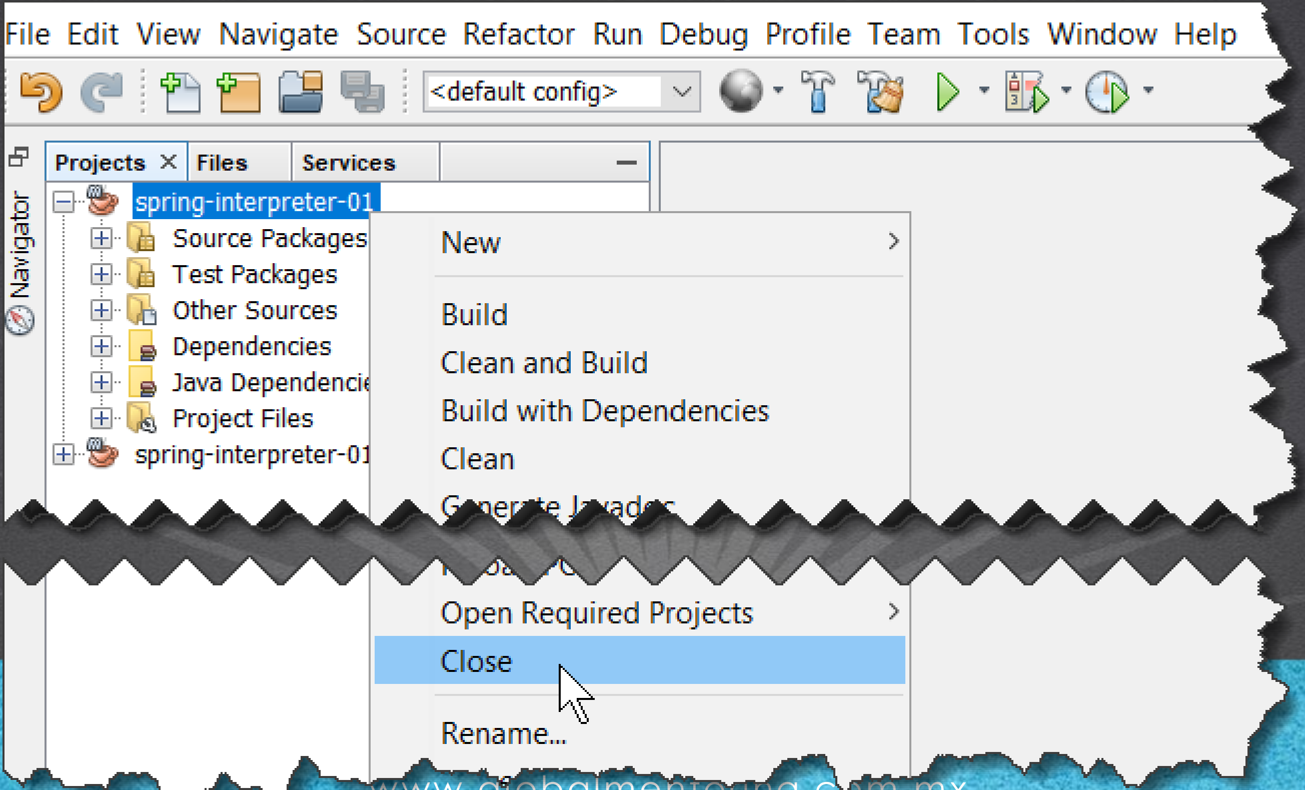
Project Location: C:\Courses\Spring\Lesson02

Project Folder: C:\Courses\Spring\Lesson02\spring-interpreter-02

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

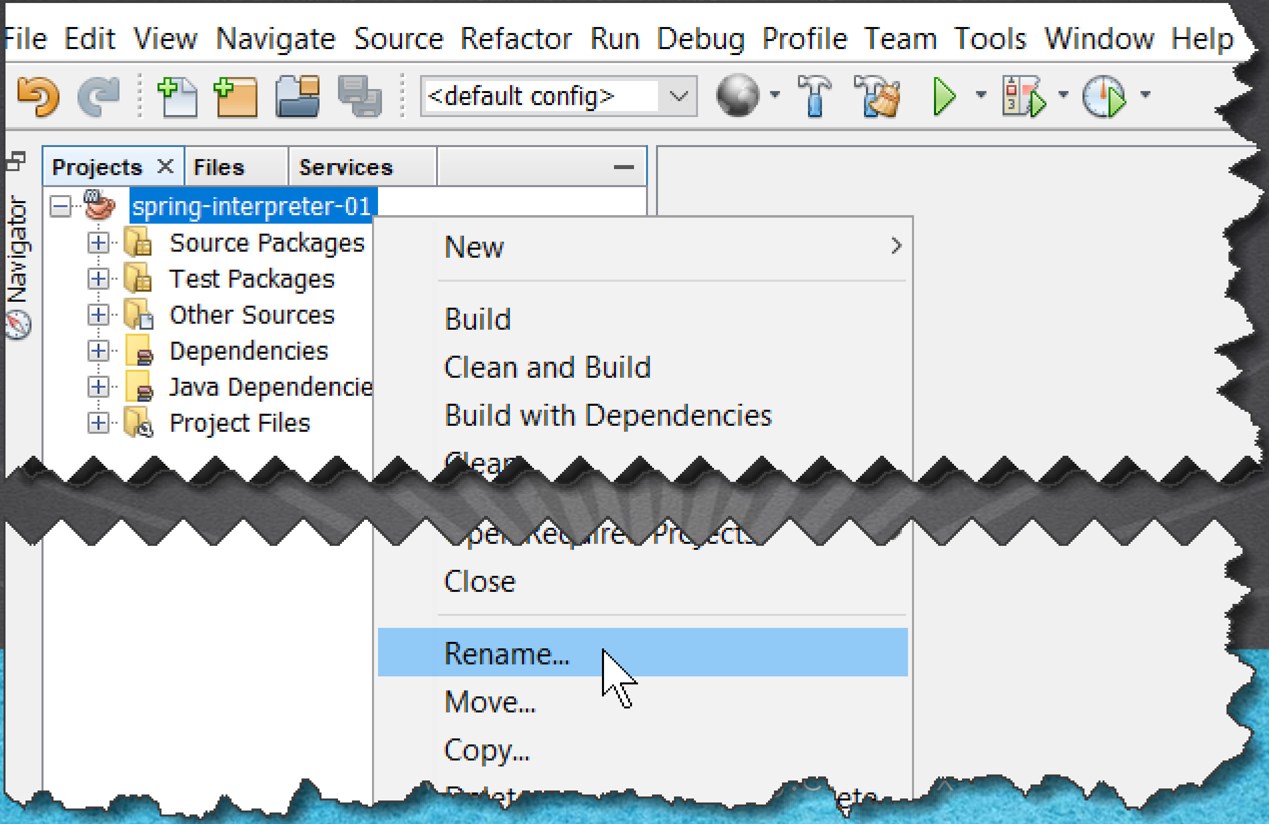
2. CLOSE THE PREVIOUS PROJECT

Close the previous Project:



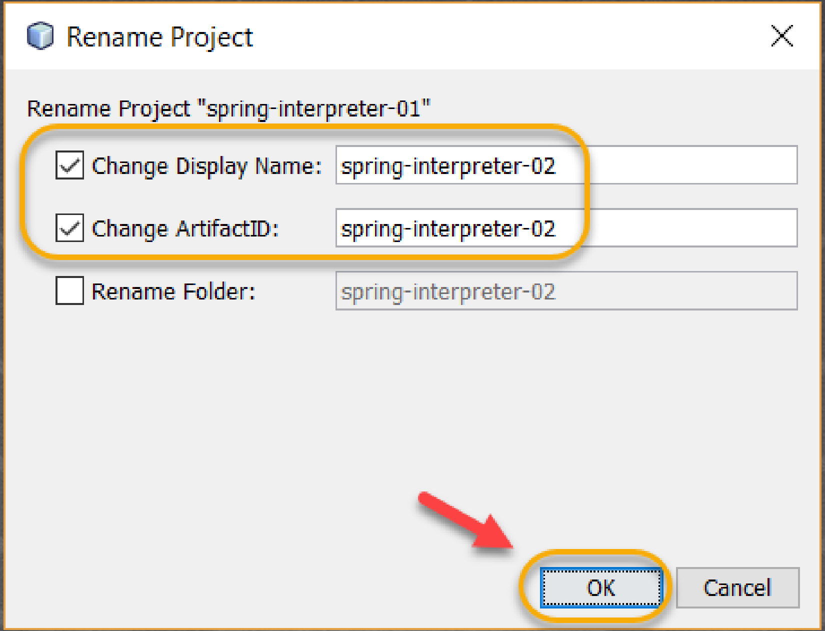
3. RENAME THE PROJECT

We rename the new project to spring-interpreter-02:



3. RENAME THE PROJECT

We rename the new project to spring-interpreter-02:



Rename Project

Rename Project "spring-interpreter-01"

☑ Change Display Name: spring-interpreter-02

☑ Change ArtifactID: spring-interpreter-02

□ Rename Folder: spring-interpreter-02

4. ADD THE FOLLOWING LIBRARIES

We add the following libraries to our project :

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

As we move forward, we will use more libraries.

4. 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-interpreter-02</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>
```

4. MODIFY THE FILE

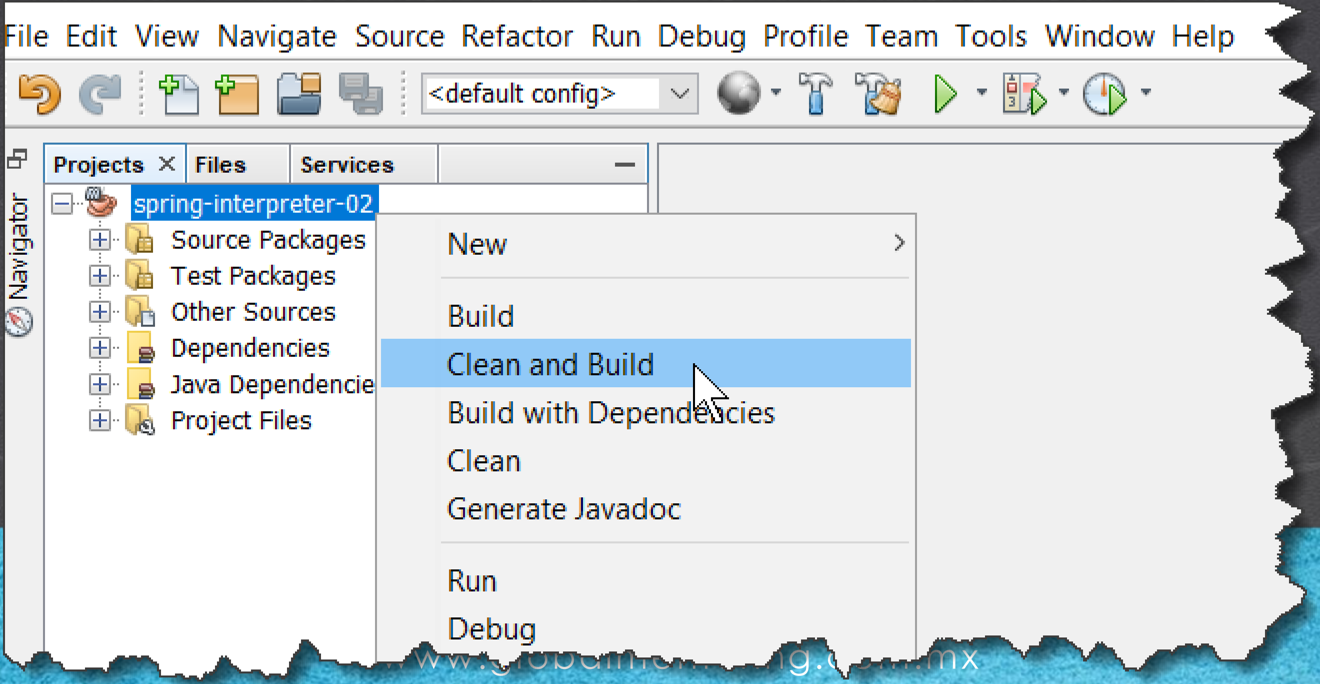
pom.xml:

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>
    <name>spring-interpreter-02</name>
</project>
```

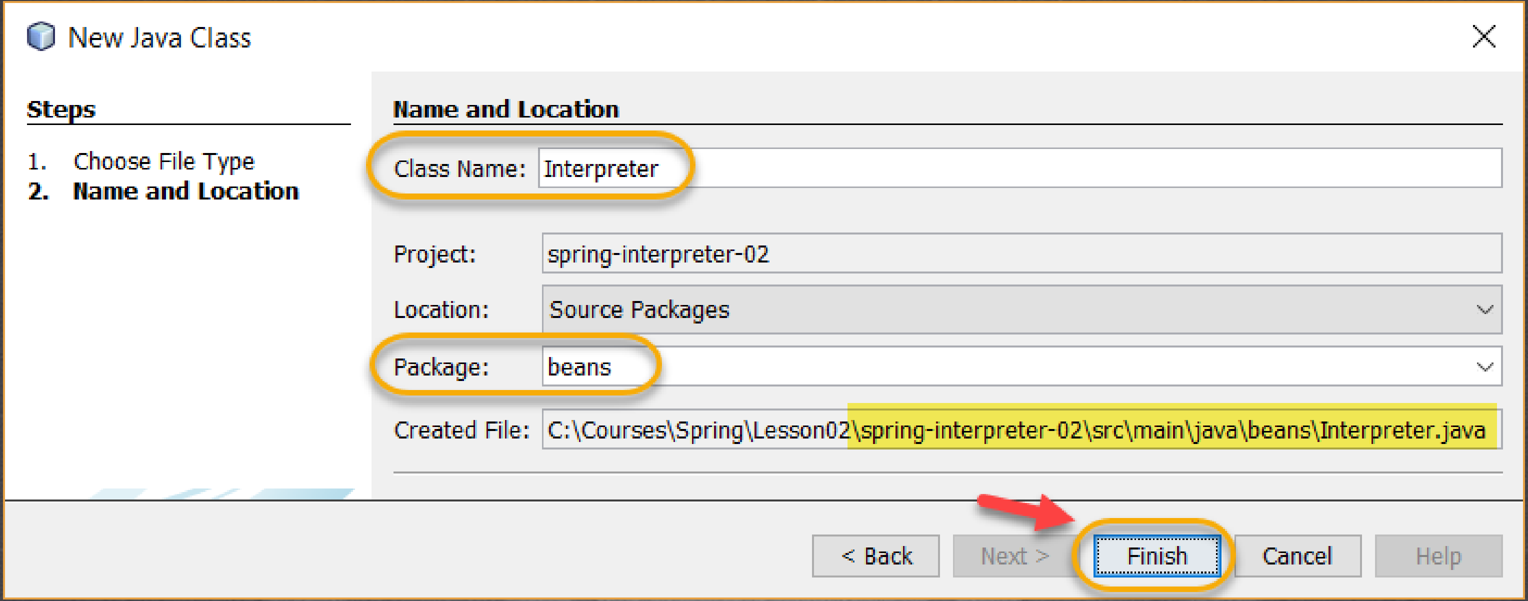
4. EXECUTE CLEAN & BUILD

At the end of adding the libraries to the pom.xml file, we do a clean & build of the project to download the missing libraries if there was a need to update the maven repository with these libraries:



5. CREATE A JAVA CLASS

Create the Interpreter.java interface:



New Java Class

Steps

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

Name and Location

Class Name: Interpreter

Project: spring-interpreter-02

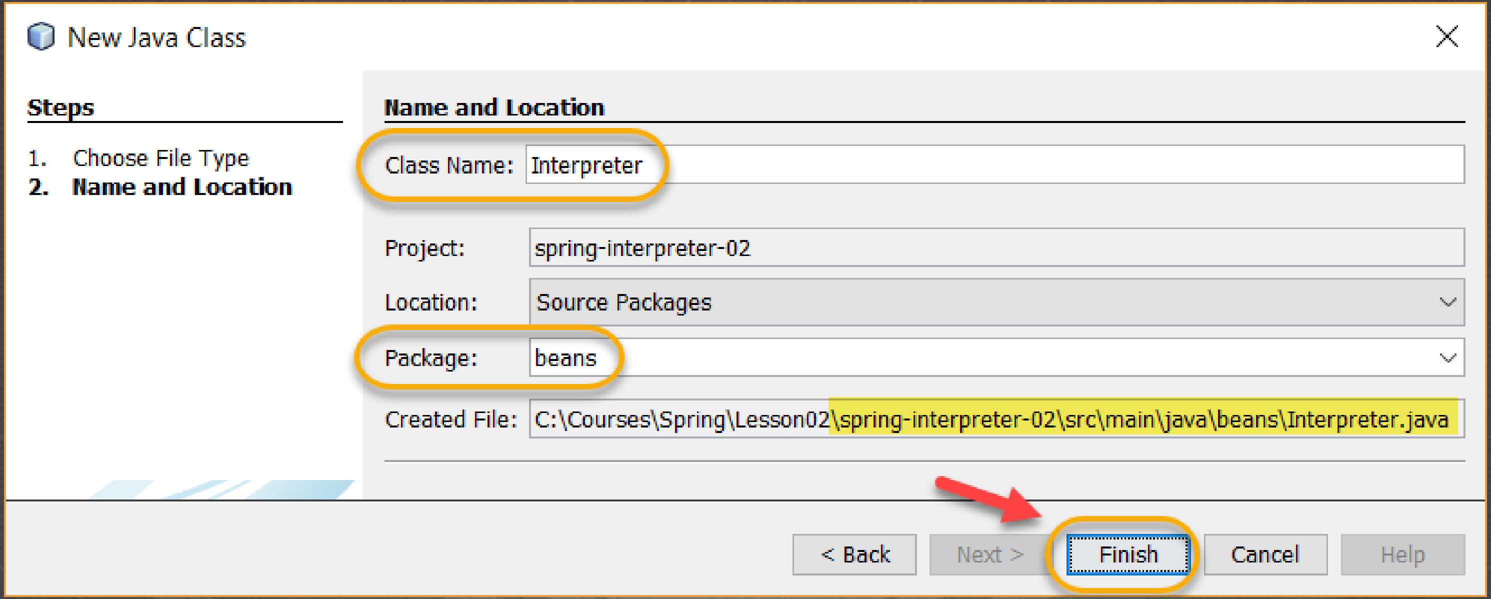
Location: Source Packages

Package: beans

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\java\beans\Interpreter.java

5. CREATE A JAVA CLASS

Create the Interpreter.java interface:



New Java Class

Steps

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

Name and Location

Class Name: Interpreter

Project: spring-interpreter-02

Location: Source Packages

Package: beans

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\java\beans\Interpreter.java

6. MODIFY THE CODE

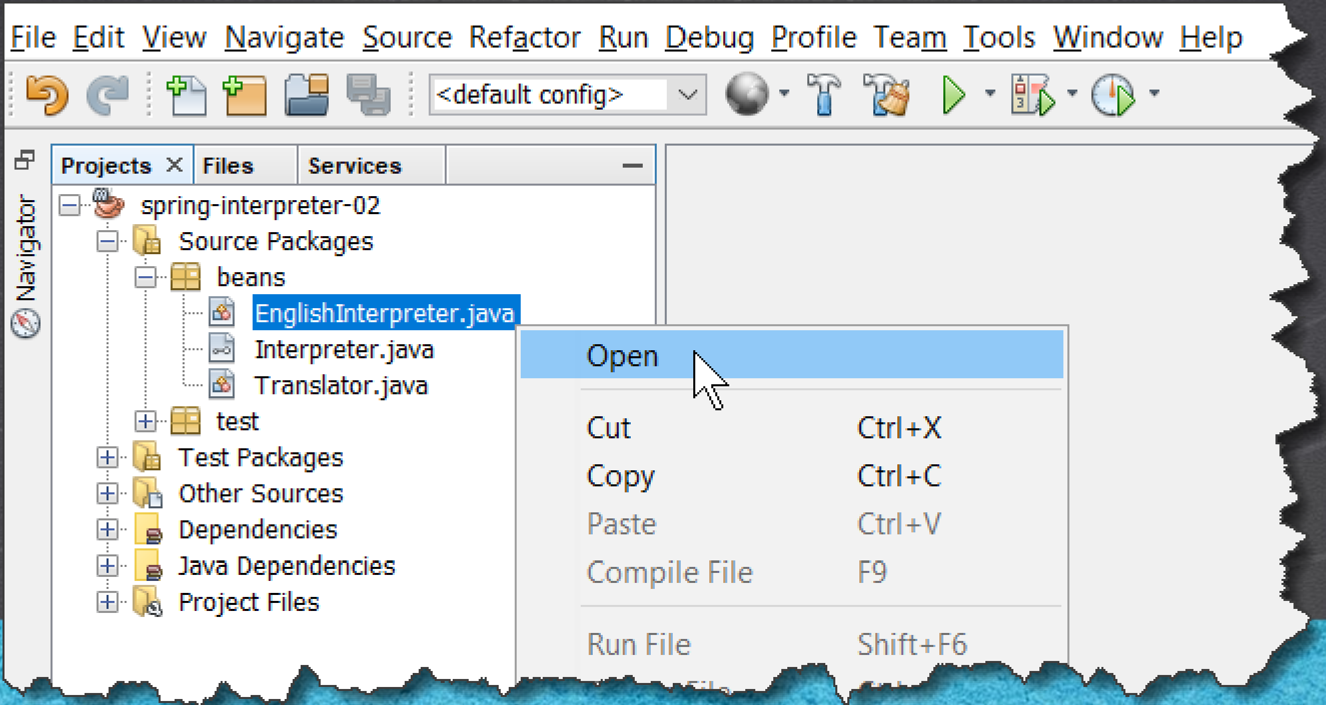
Interpreter.java:

Click to download

```
package beans;

public interface Interpreter {

    public void sayHello();

    public void sayGoodbye();
}
```

7. MODIFY THE CLASS

Modify the EnglishInterpreter.java to implement the interface:



7. MODIFY THE CODE

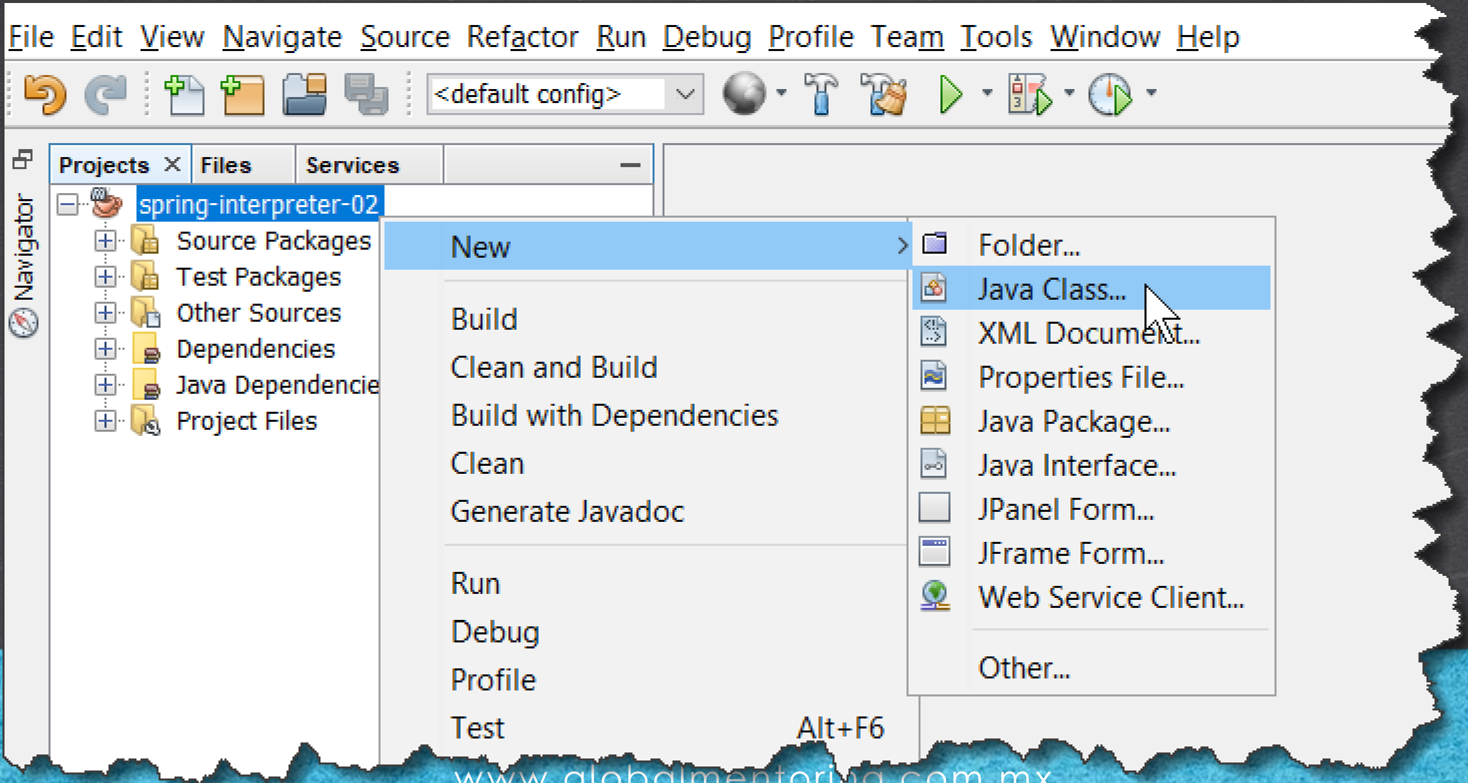
EnglishInterpreter.java:

Click to download

```java
package beans;

public class EnglishInterpreter implements Interpreter{

    @Override
    public void sayHello() {
        System.out.println("Hello, my name is:");
    }

    @Override
    public void sayGoodbye() {
        System.out.println("See you soon...");
    }
}
```

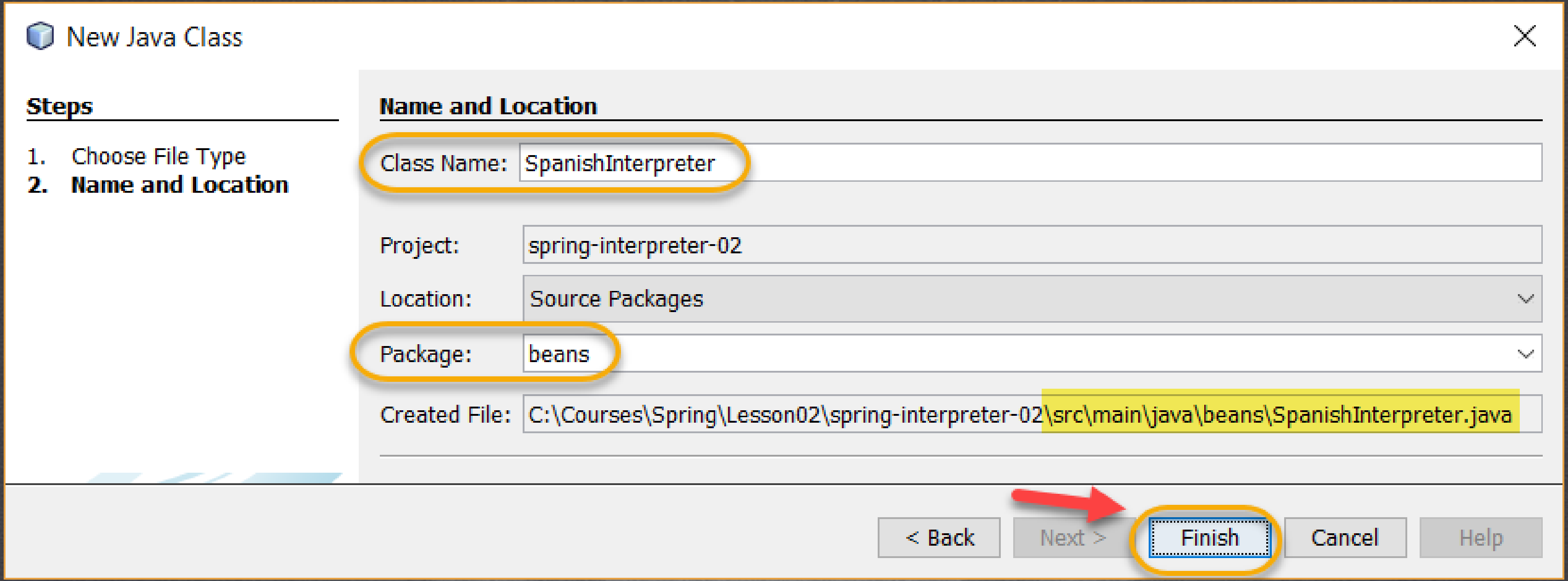
8. CREATE A JAVA CLASS

We create the SpanishInterpreter.java class:



8. CREATE A NEW CLASS

We create the SpanishInterpreter.java class:



New Java Class

Steps

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

Name and Location

Class Name: SpanishInterpreter

Project: spring-interpreter-02

Location: Source Packages

Package: beans

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\java\beans\SpanishInterpreter.java

9. MODIFY THE FILE

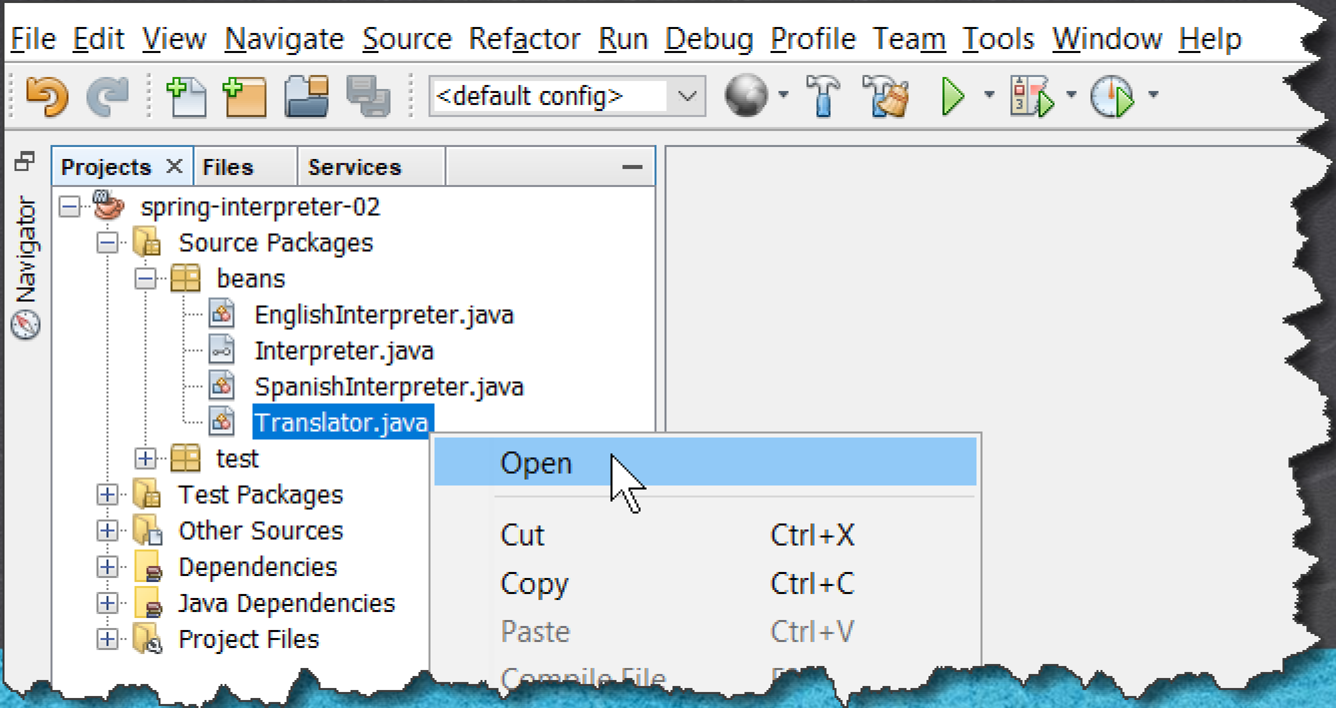
EnglishInterpreter.java:

Click to download

```java
package beans;

public class EnglishInterpreter implements Interpreter{

    @Override
    public void sayHello() {
        System.out.println("Hello, my name is:");
    }

    @Override
    public void sayGoodbye() {
        System.out.println("See you soon...");
    }
}
```

10. MODIFY THE JAVA CLASS

Modify the Translator class in order to use the interface type:



10. MODIFY THE FILE

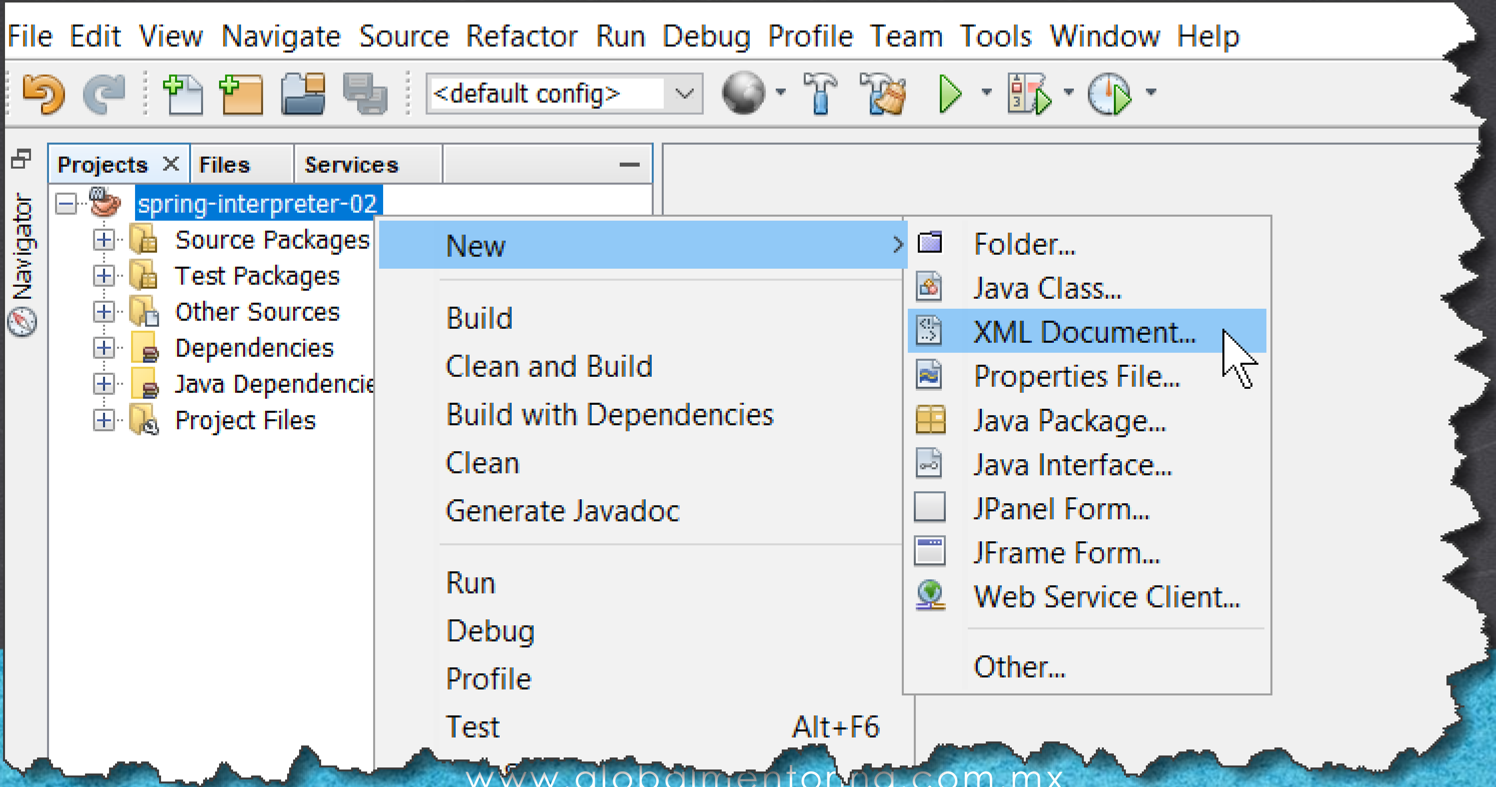
Translator.java:

Click to download

```
package beans;

public class Translator {

    private Interpreter interpreter;
    private String name;

    public void speak() {
        this.interpreter.sayHello();
        System.out.println(name);
        this.interpreter.sayGoodbye();
    }

    public Interpreter getInterpreter() {
        return interpreter;
    }

    public void setInterpreter(Interpreter interpreter) {
        this.interpreter = interpreter;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

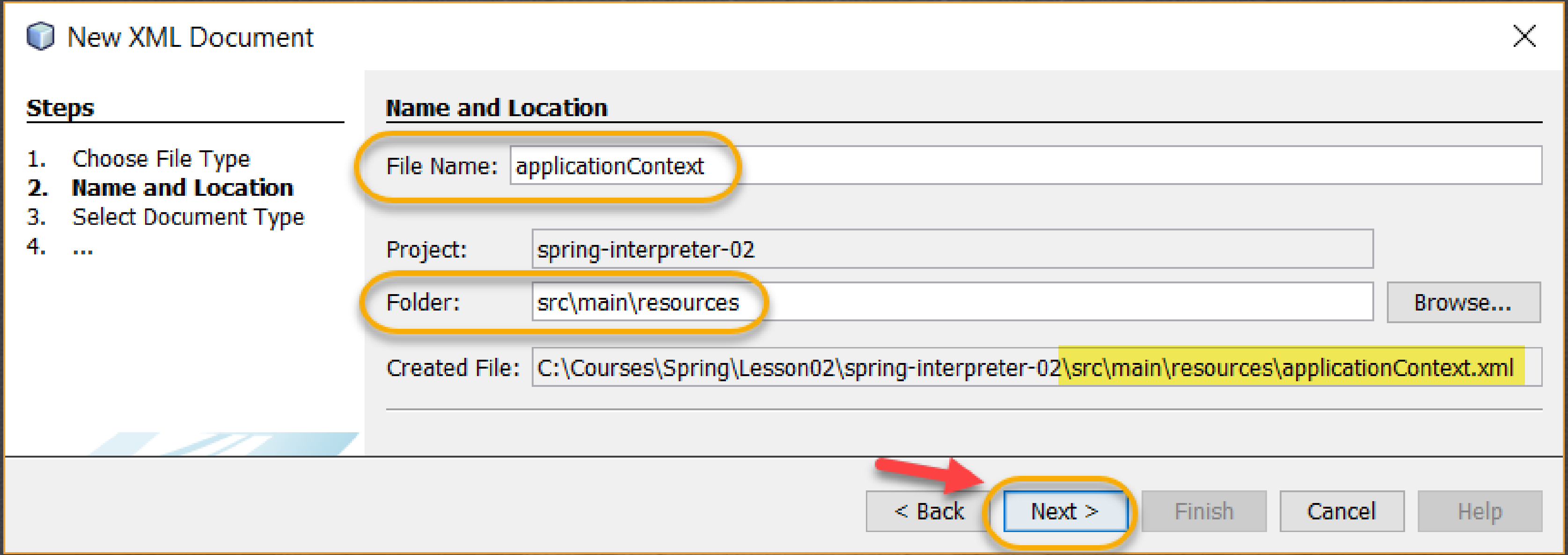
11. CREATE AN XML FILE

We create the applicationContext.xml file:



11. CREATE AN XML FILE

We create the applicationContext.xml file:



New XML Document

Steps

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

Name and Location

File Name: applicationContext

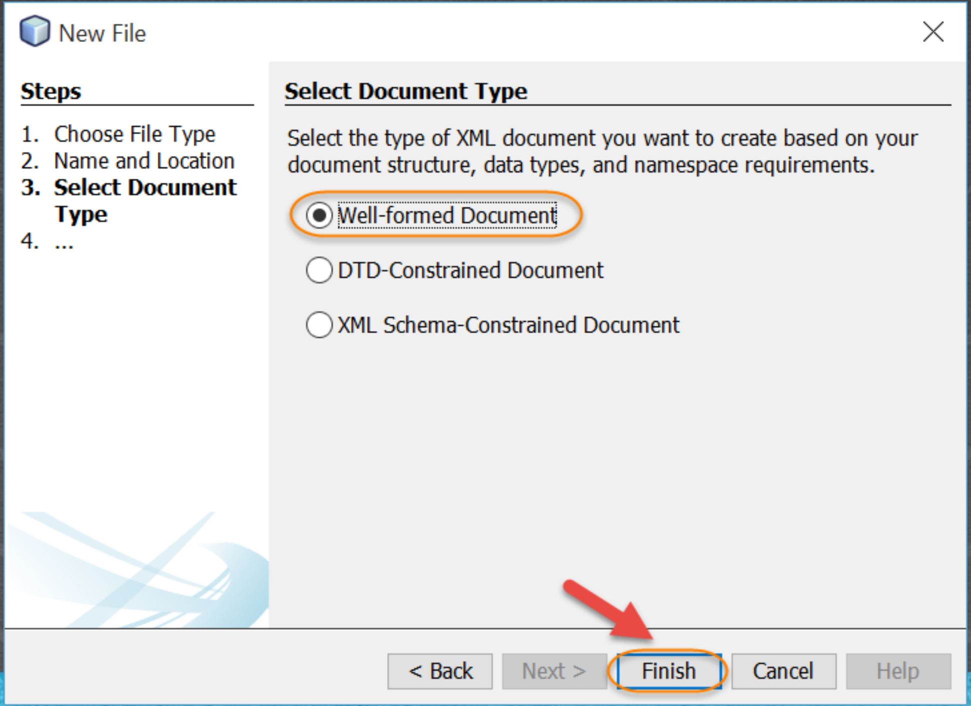
Project: spring-interpreter-02

Folder: src\main\resources

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\resources\applicationContext.xml

11. CREATE AN XML FILE

We create the applicationContext.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

12. MODIFY THE FILE

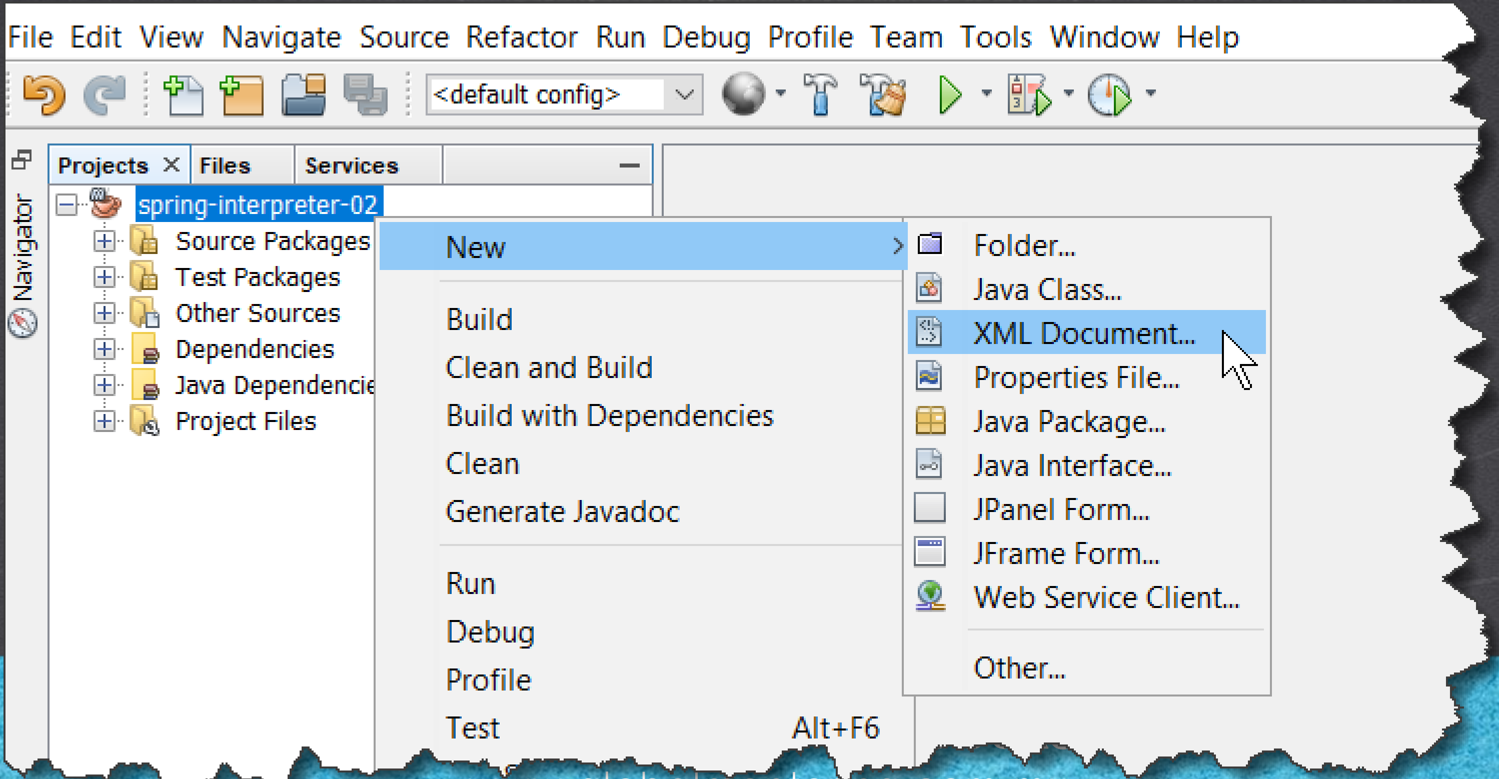
applicationContext.xml:

Click to download

```
<?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">
    <!-- we define at the beginning the less dependent beans-->
    <!-- Equivalent to the Java code:
        EnglishInterpreter englishInterpreter = new EnglishInterpreter();
    -->
    <bean id="englishInterpreter" class="beans.EnglishInterpreter"/>
    <bean id="spanishInterpreter" class="beans.SpanishInterpreter"/>

    <bean id="englishTranslator" class="beans.Translator">
          <!-- equivalent to: translator.setInterprete (englishInterpreter); -->
        <property name="interpreter" ref="englishInterpreter"/>
        <property name="name" value="Charly" />
    </bean>

    <!-- we define the most dependent beans at the end -->
    <bean id="spanishTranslator" class="beans.Translator">
        <!-- We can add a different interpreter to the translator class
        because we use an interface type Interpreter  -->
        <property name="interpreter" ref="spanishInterpreter"/>
        <!-- equivalent to: translator.setName ("Juan"); -->
        <property name="name" value="Juan"/>
    </bean>
</beans>
```

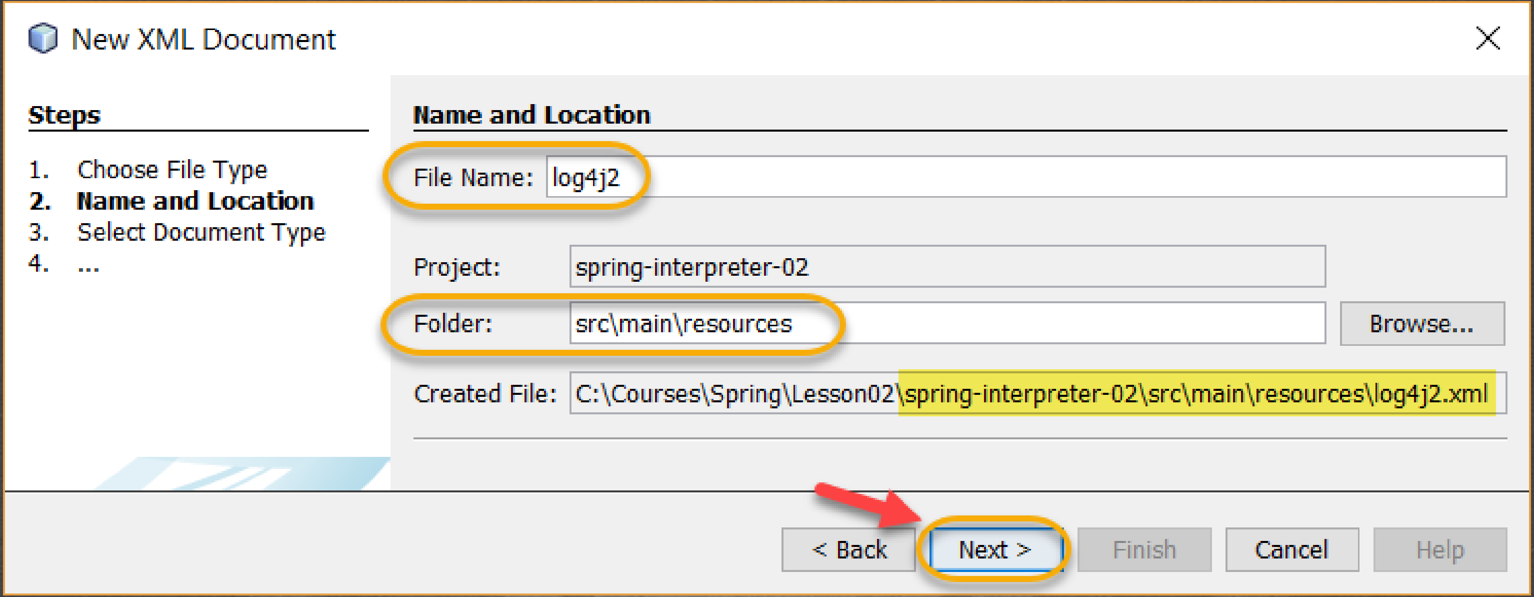
13. CREATE AN XML FILE

We create the log4j2.xml file:



13. 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

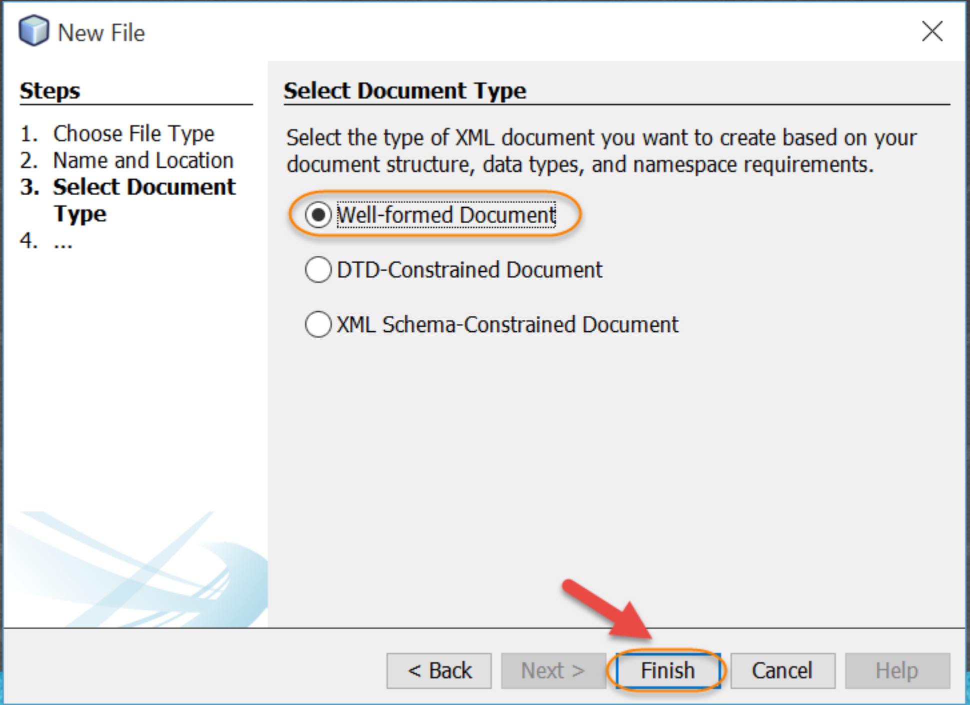
Project: spring-interpreter-02

Folder: src\main\resources

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\resources\log4j2.xml

13. CREATE AN XML FILE

We create the log4j2.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

14. MODIFY THE FILE

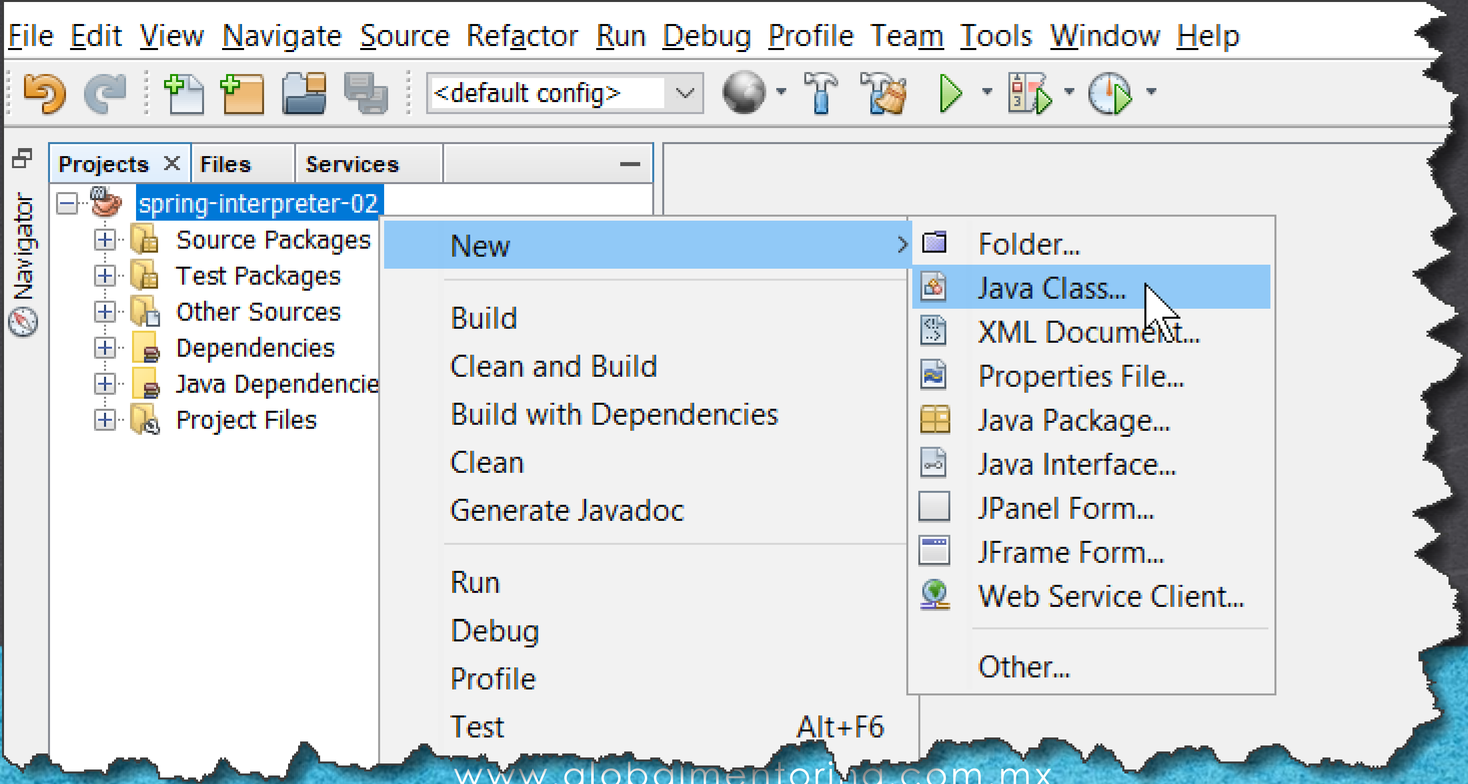
log4j2.xml:

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>
```

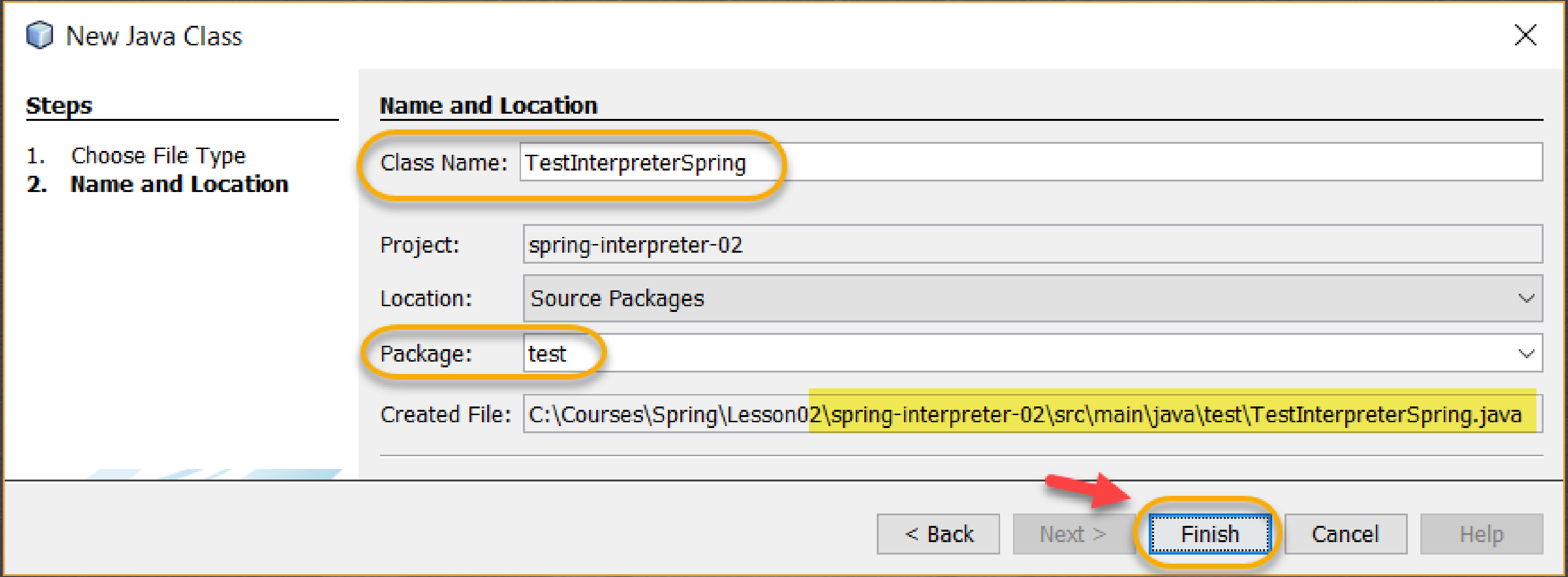
15. CREATE A CLASS

Create the TestInterpreterSpring.java class:



15. CREATE A JAVA CLASS

Create the TestInterpreterSpring.java class:



New Java Class

Steps

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

Name and Location

Class Name: TestInterpreterSpring

Project: spring-interpreter-02

Location: Source Packages

Package: test

Created File: C:\Courses\Spring\Lesson02\spring-interpreter-02\src\main\java\test\TestInterpreterSpring.java

16. MODIFY THE CODE

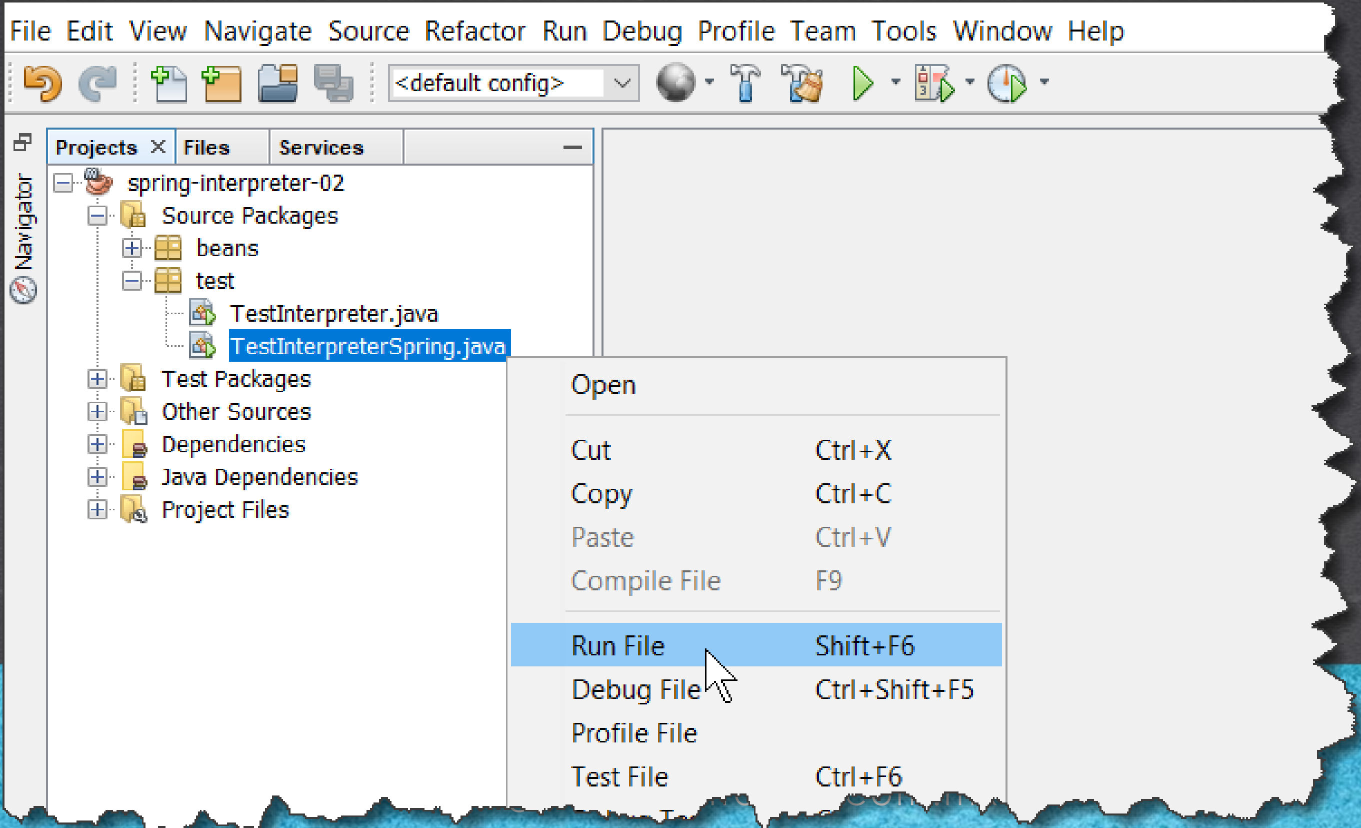
TestInterpreterSpring.java:

Click to download

```java
package test;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import beans.Translator;

public class TestInterpreterSpring {

    public static void main(String[] args) {
        BeanFactory factory = new ClassPathXmlApplicationContext("applicationContext.xml");

        Translator translator1 =  (Translator) factory.getBean("englishTranslator");
        translator1.speak();

        System.out.println();

        Translator translator2 = (Translator) factory.getBean("spanishTranslator");
        translator2.speak();
    }
}
```

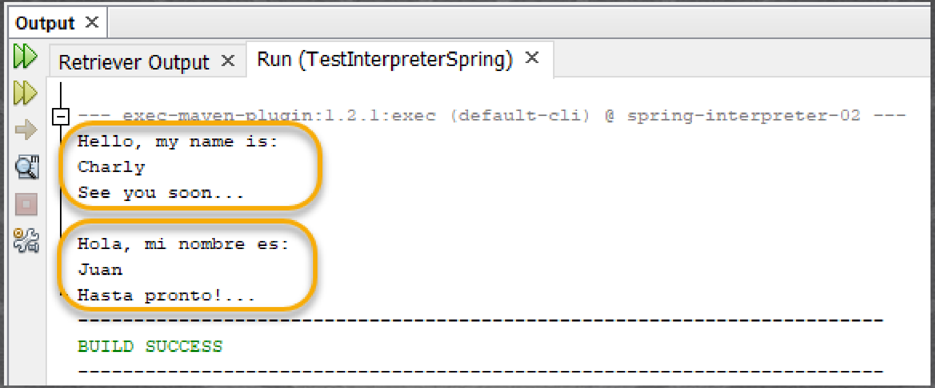
17. EXECUTE THE PROJECT

Execute the TestInterpreterSpring.java class:



17. EXECUTE THE PROJECT

The output of the program is as follows. We see the result of executing both interpreters, both in English and in Spanish, which were injected as dependencies from the Spring Framework applicationContext.xml file:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ spring-interpreter-02 ---
Hello, my name is:
Charly
See you soon...

Hola, mi nombre es:
Juan
Hasta pronto!...
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

EXERCISE CONCLUSION

With this exercise we were able to observe that we already obtain the translators beans ready to be used, and we did not have to directly handle the dependencies, but rather Spring manages these dependencies.

This is the concept of dependency injection, which is one of the concepts that we will most use when using the Spring framework.

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta

Global Mentoring

SPRING FRAMEWORK COURSE

www.globalmentoring.com.mx