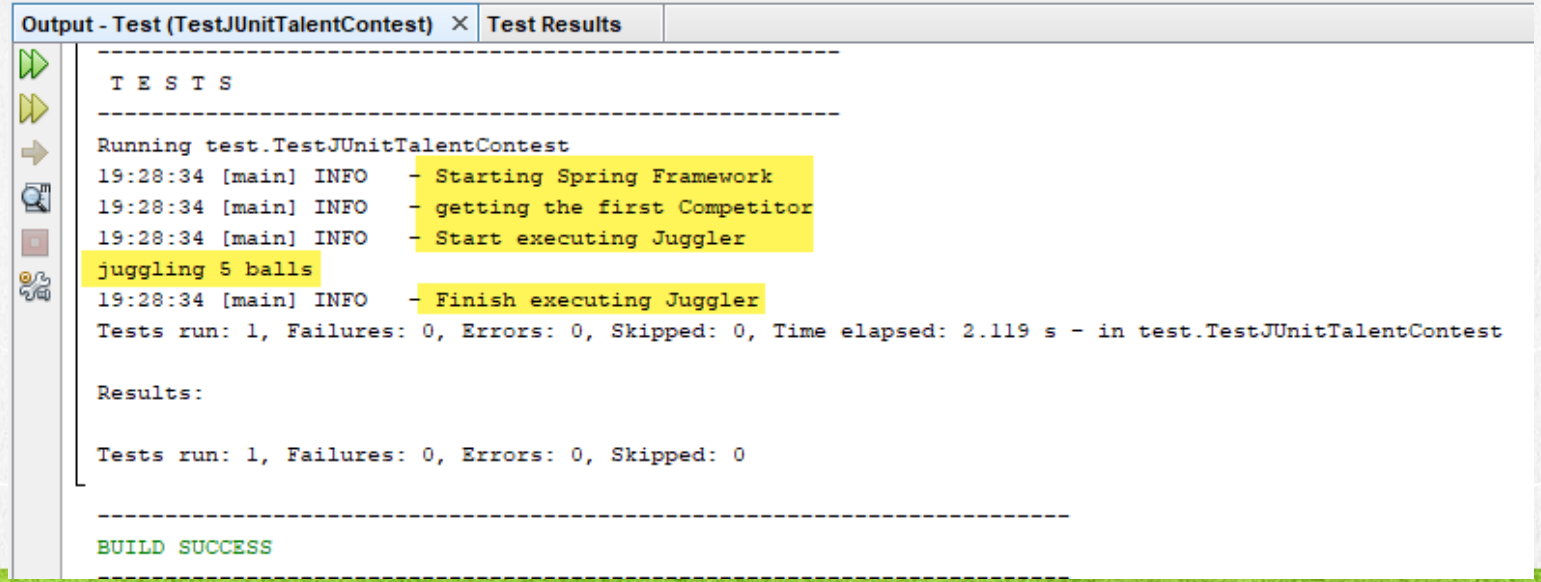
SPRING FRAMEWORK COURSE

EXERCISE

TALENT CONTEST V1 WITH SPRING FRAMEWORK

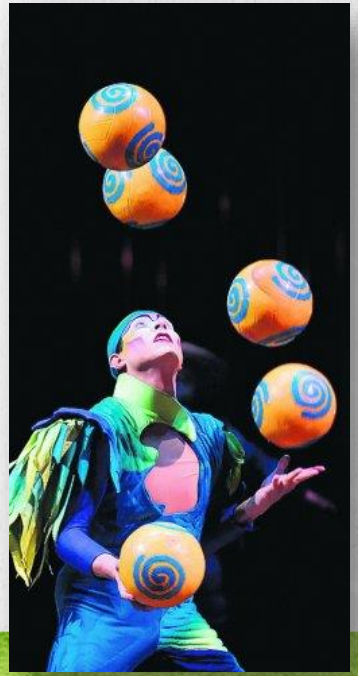
EXERCISE OBJECTIVE

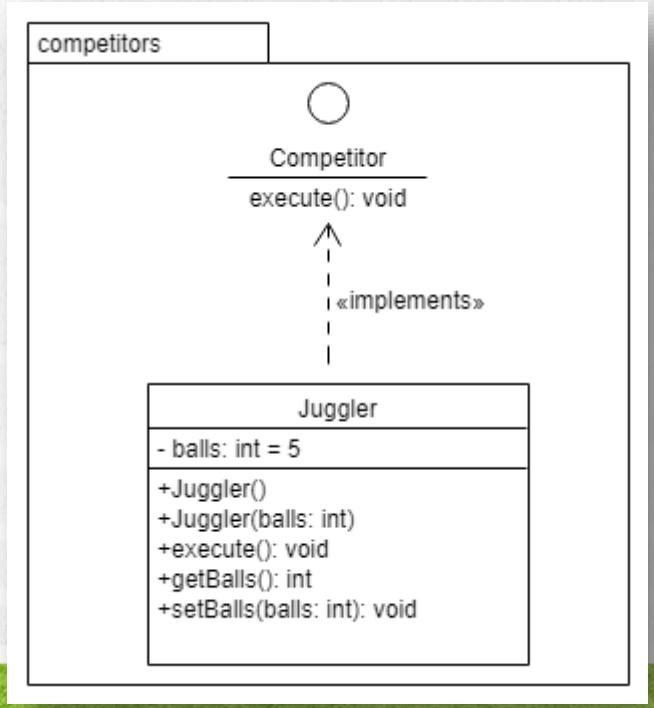
The objective of the exercise is to create the Talent Contest project. At the end we must observe the following output:



TALENT CONTEST PROJECT

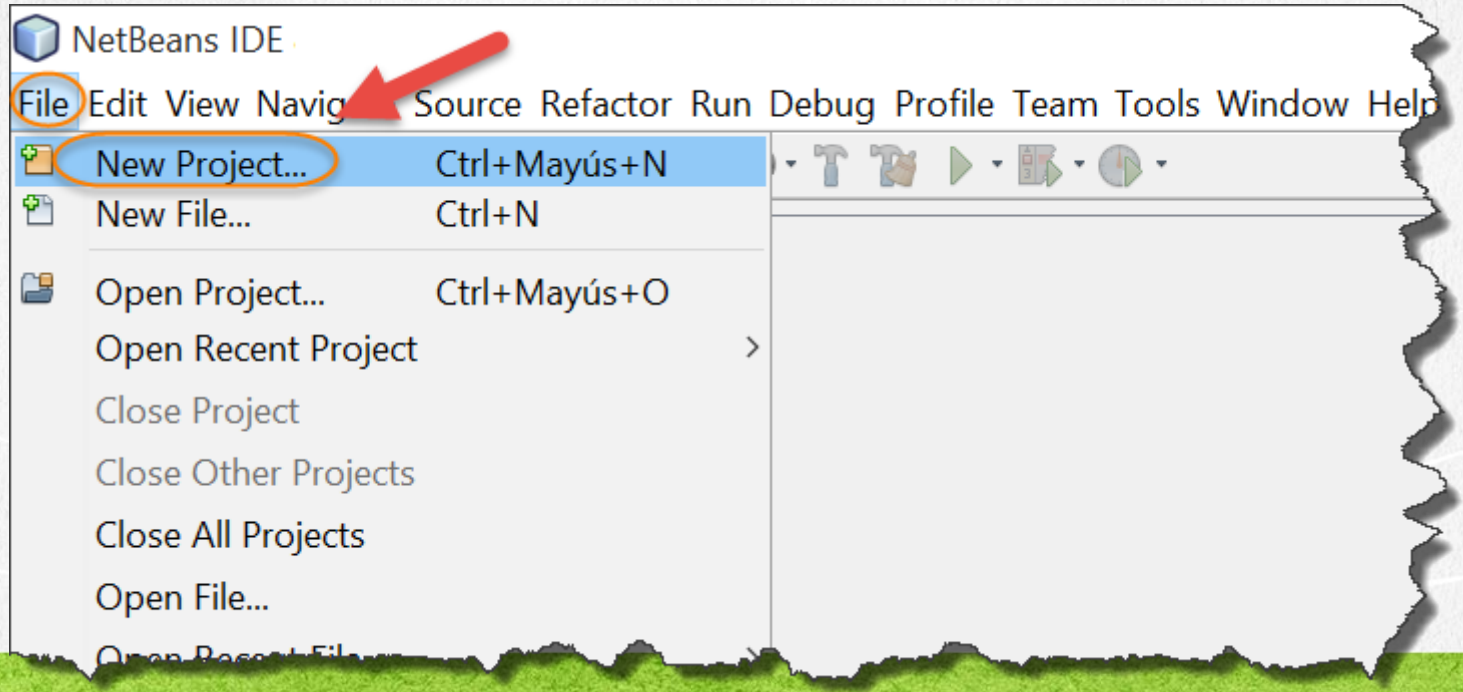
- We will create a project that contains the following classes:





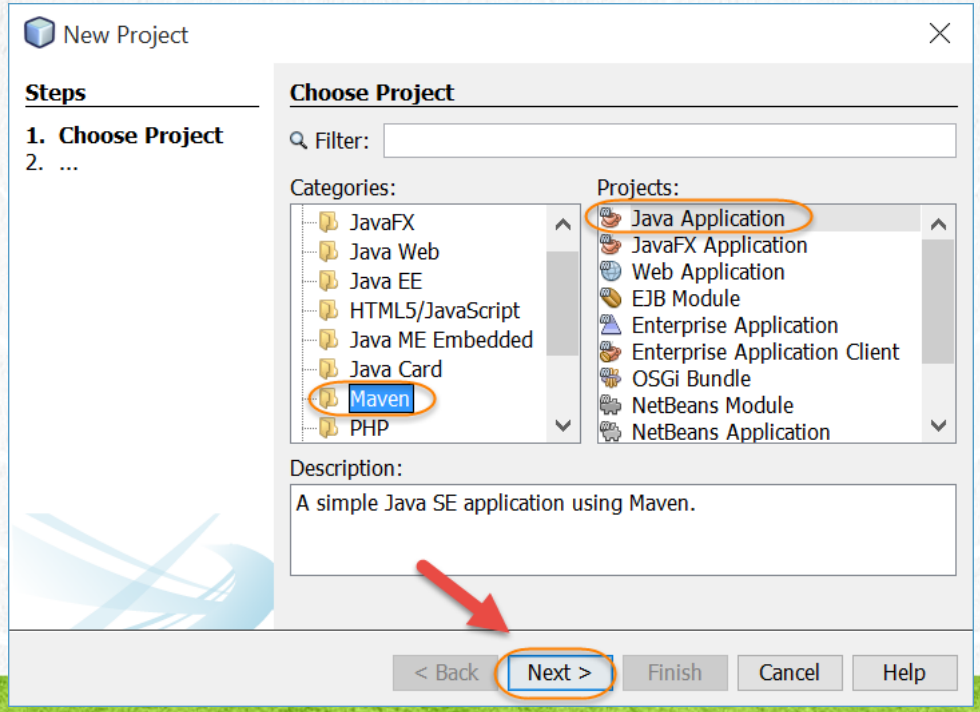
1. CREATE A NEW PROJECT

We created the project spring-talent-contest-01:



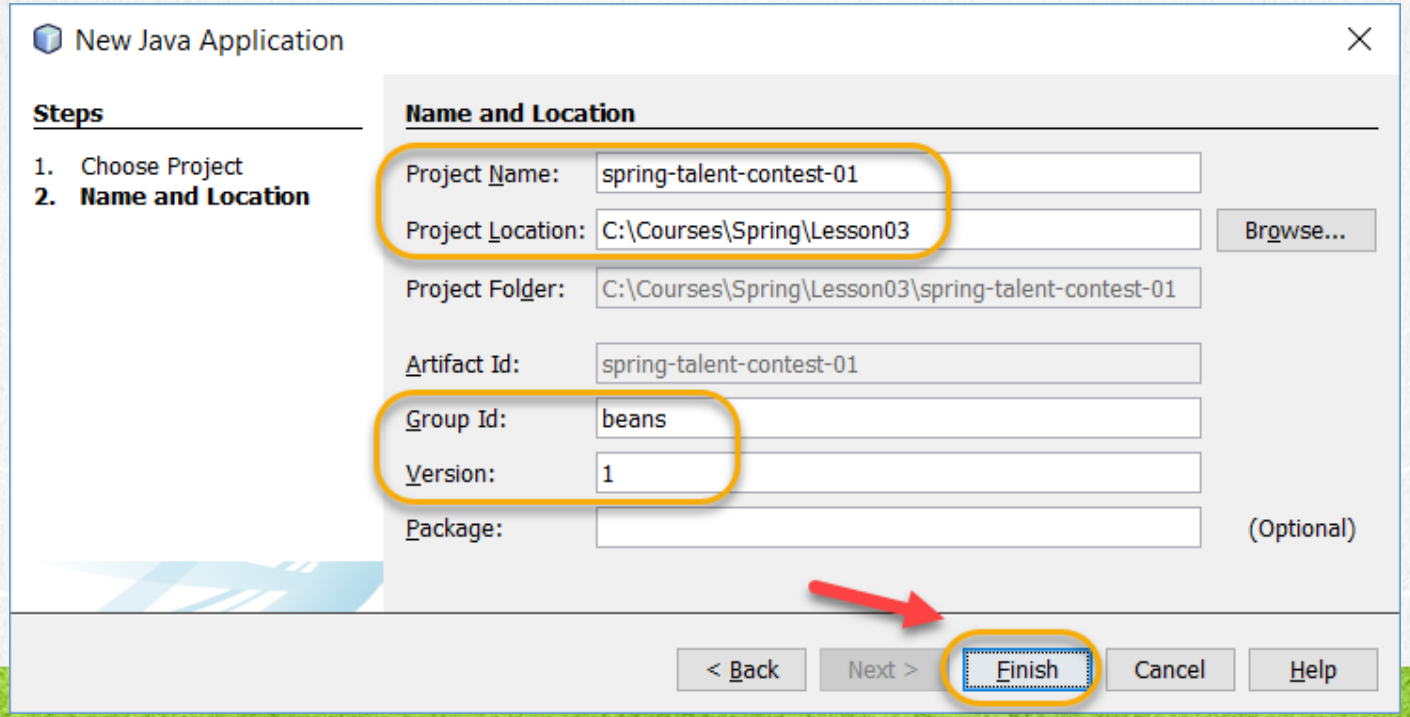
1. CREATE A NEW PROJECT

We created the project spring-talent-contest-01:



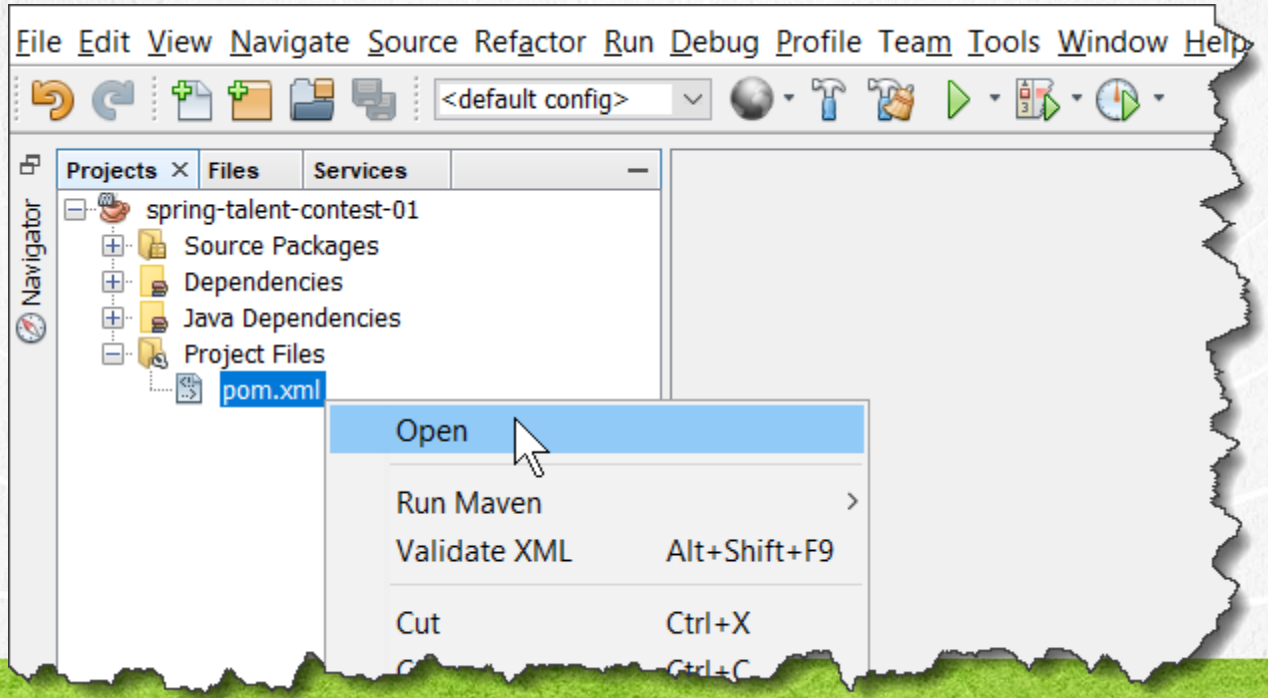
1. CREATE A NEW PROJECT

We created the project spring-talent-contest-01:



2. ADD THE LIBRARIES

We open the file pom.xml and add the libraries that we will use:



2. MODIFY THE CODE

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-talent-contest-01</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>
        <junit.version>5.3.1</junit.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 CODE

pom.xml:

Click to download

```
        <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>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

2. MODIFY THE CODE

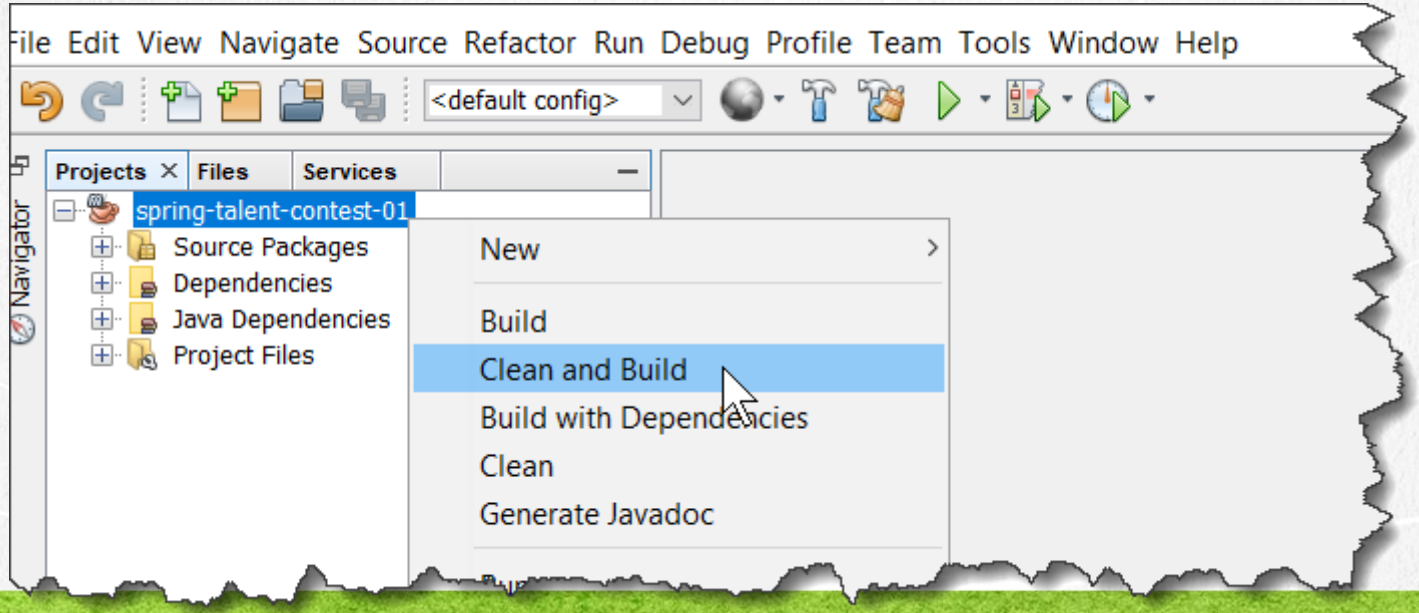
pom.xml:

Click to download

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.0</version>
            </plugin>
        </plugins>
    </build>
</project>
```

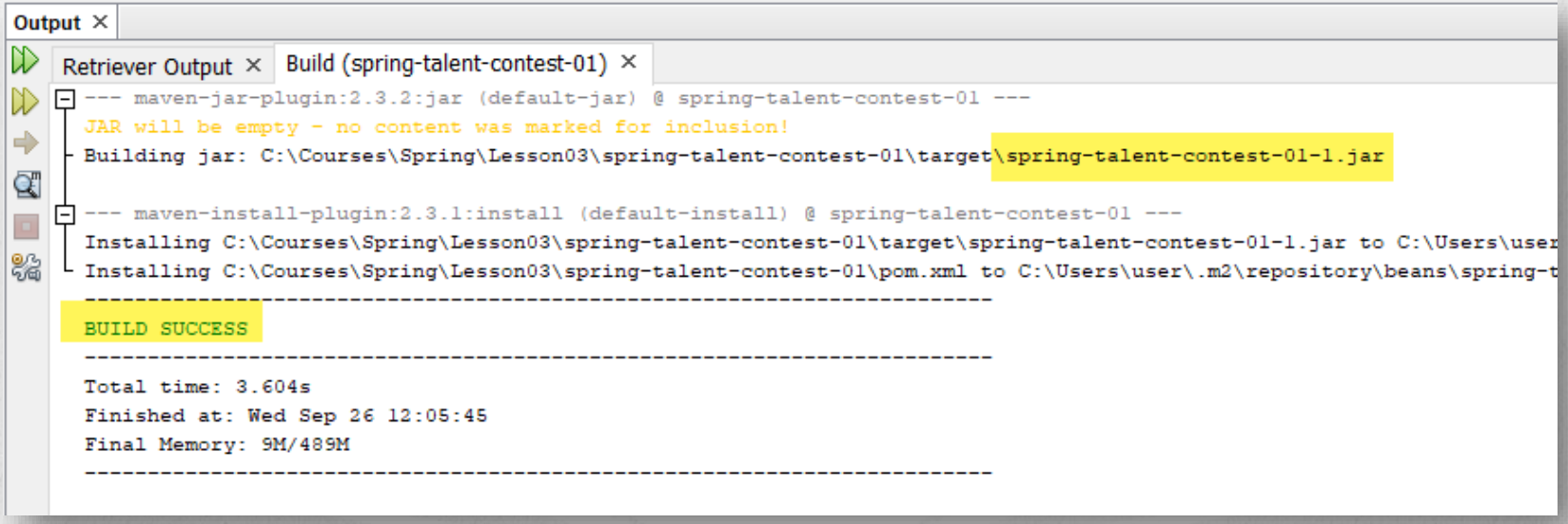
3. EXECUTE CLEAN & BUILD

We do a clean & build to the project so that the libraries are downloaded if necessary:



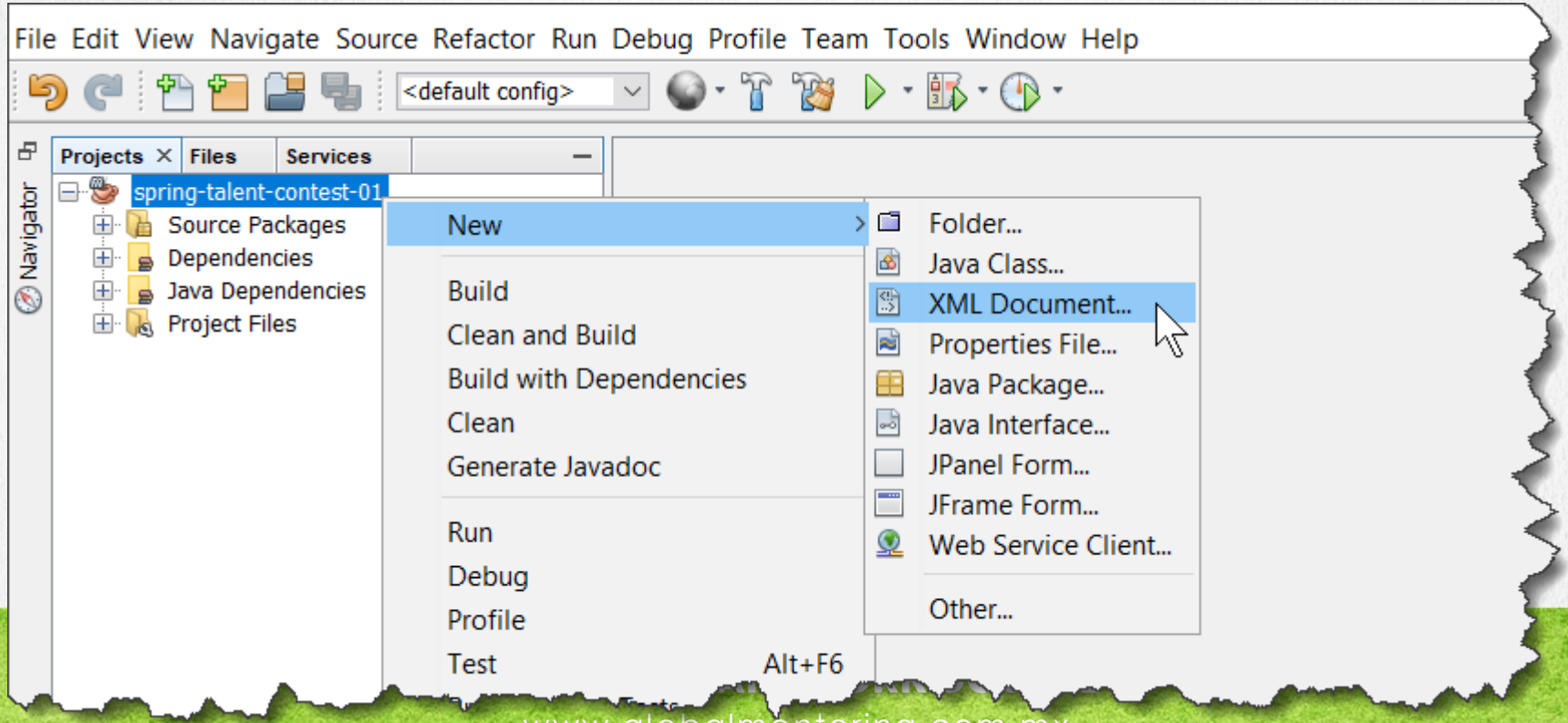
3. EXECUTE CLEAN & BUILD

We do a clean & build to the project so that the libraries are downloaded if necessary:



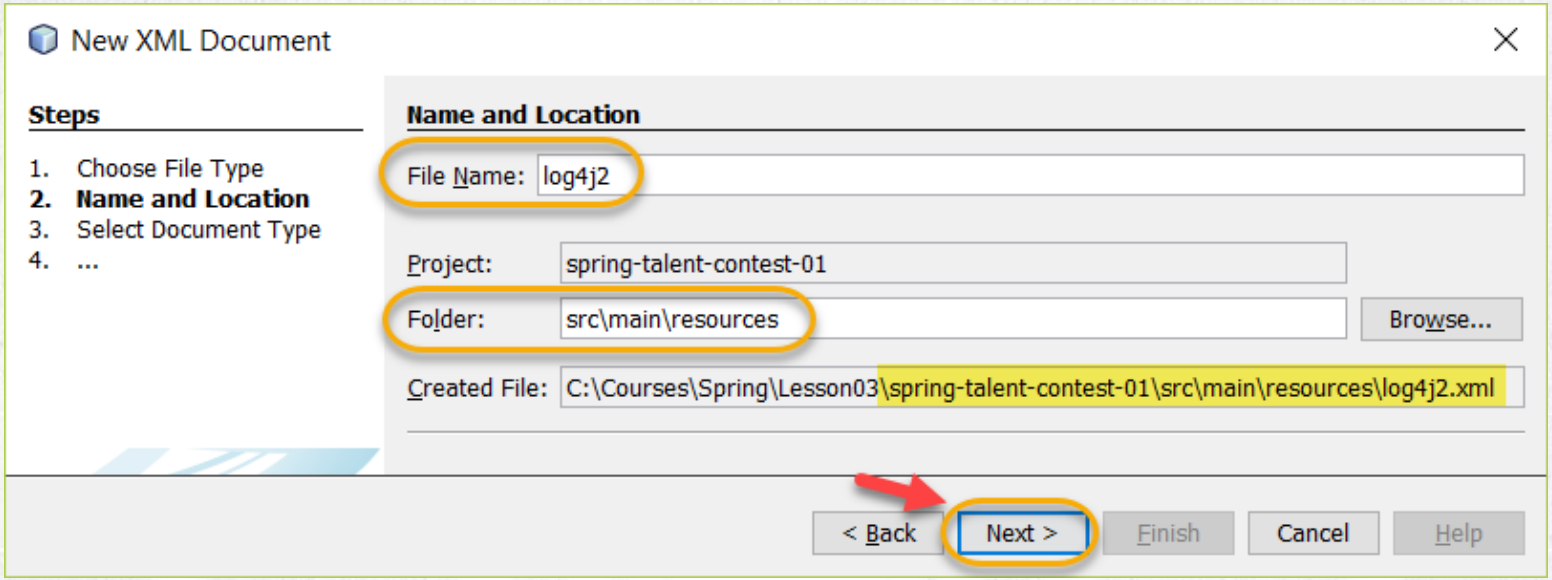
4. CREATE AN XML FILE

We create the log4j2.xml file:



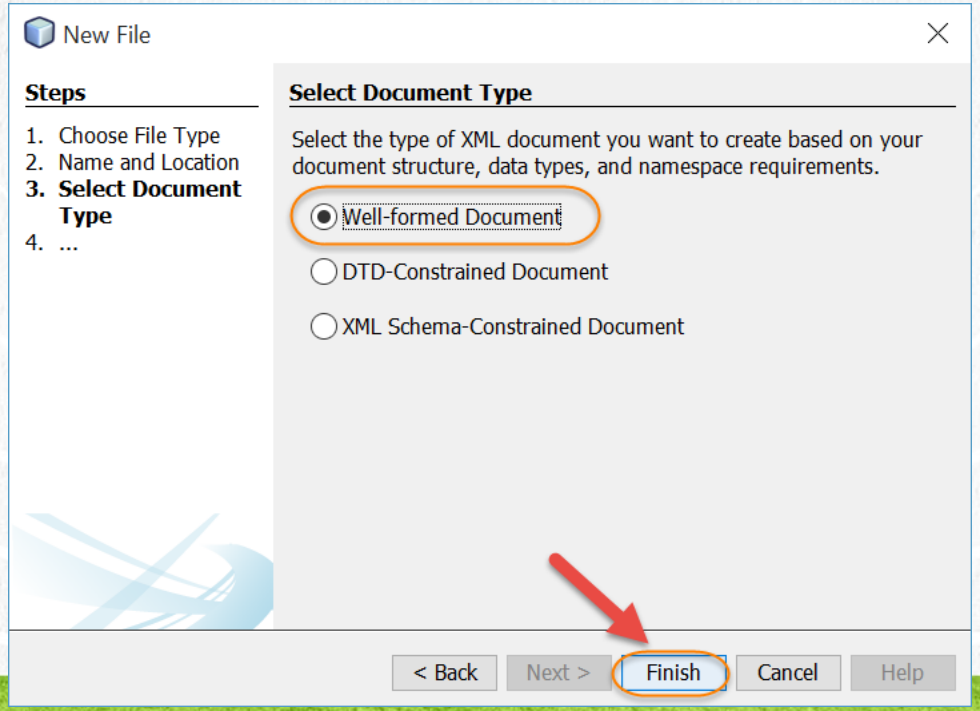
4. CREATE AN XML FILE

We create the log4j2.xml file:



4. CREATE AN XML FILE

We create the log4j2.xml file:



5. MODIFY THE FILE

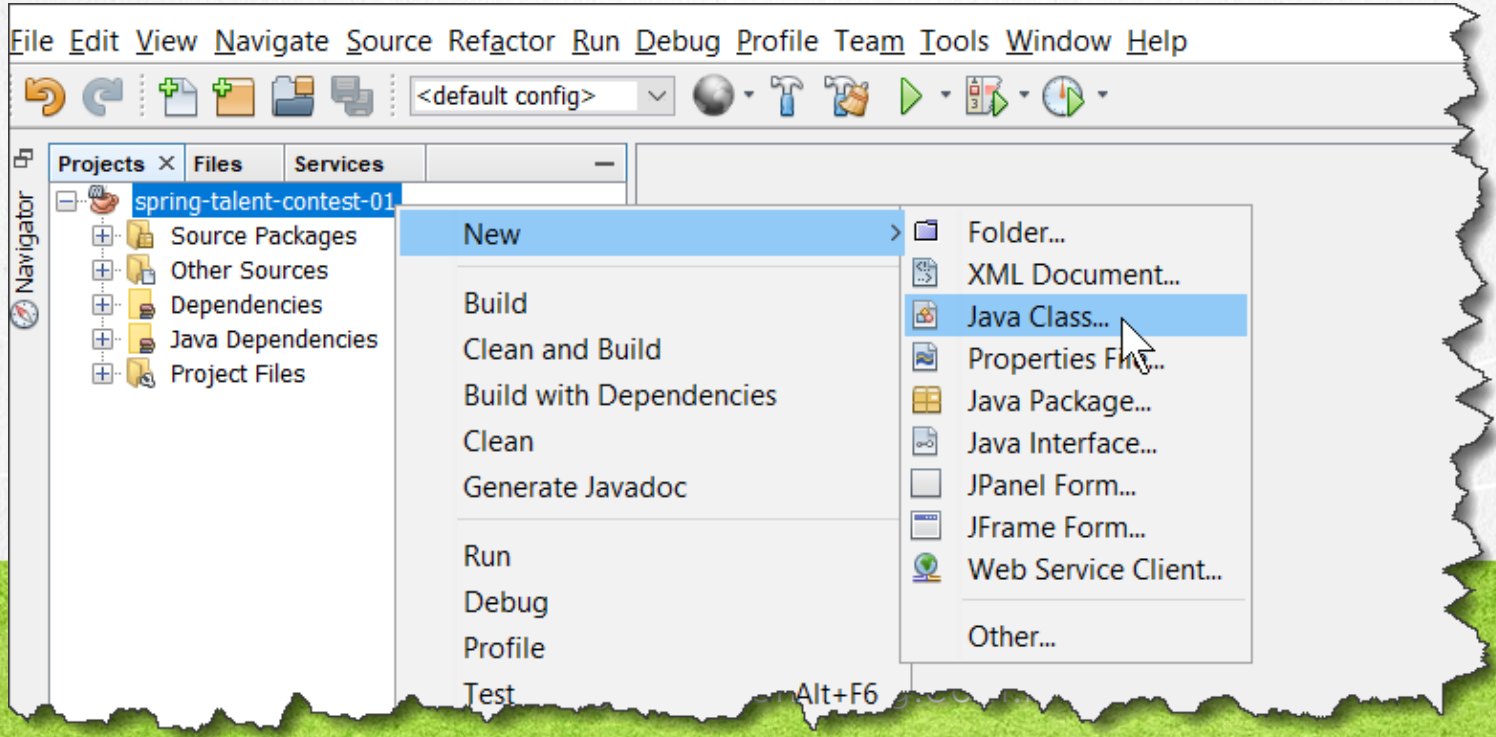
log4j2.xml:

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>
```

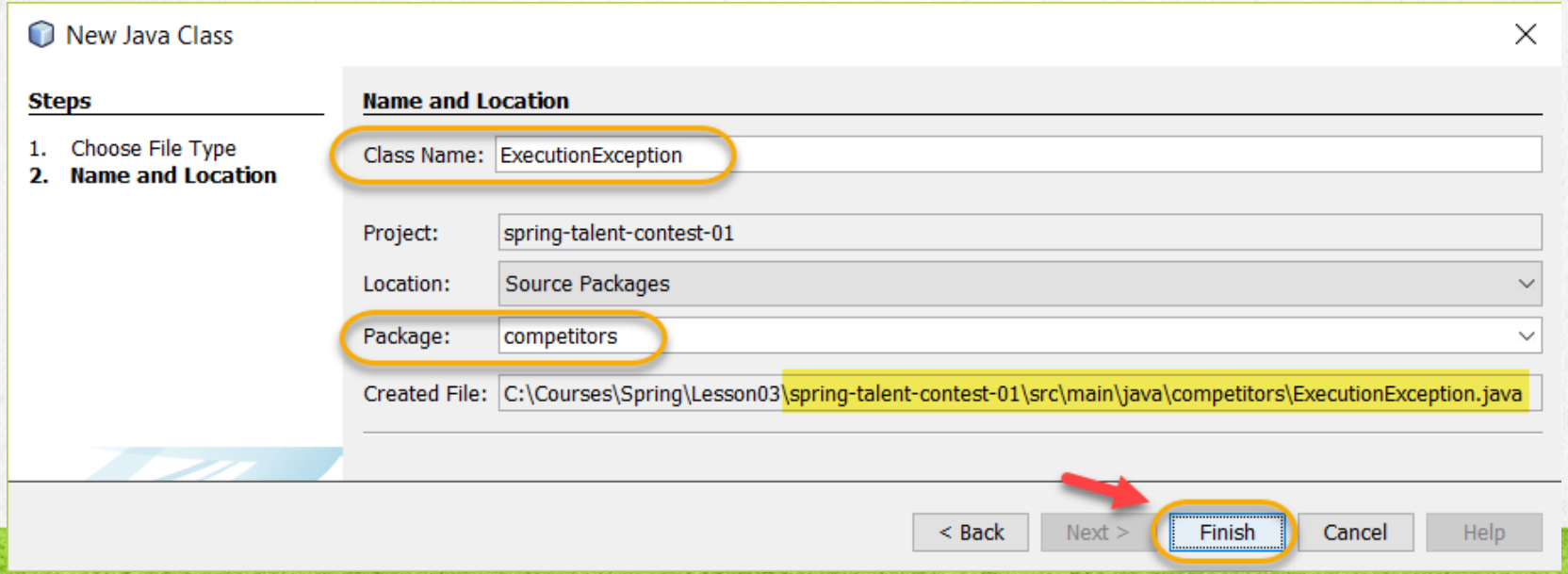
6. CREATE A JAVA CLASS

We create the ExecutionException.java class. Note: This class is used to indicate any exception, however it is not part of the domain classes of this project.



6. CREATE A JAVA CLASS

We create the ExecutionException.java class. Note: This class is used to indicate any exception, however it is not part of the domain classes of this project.



7. MODIFY THE FILE

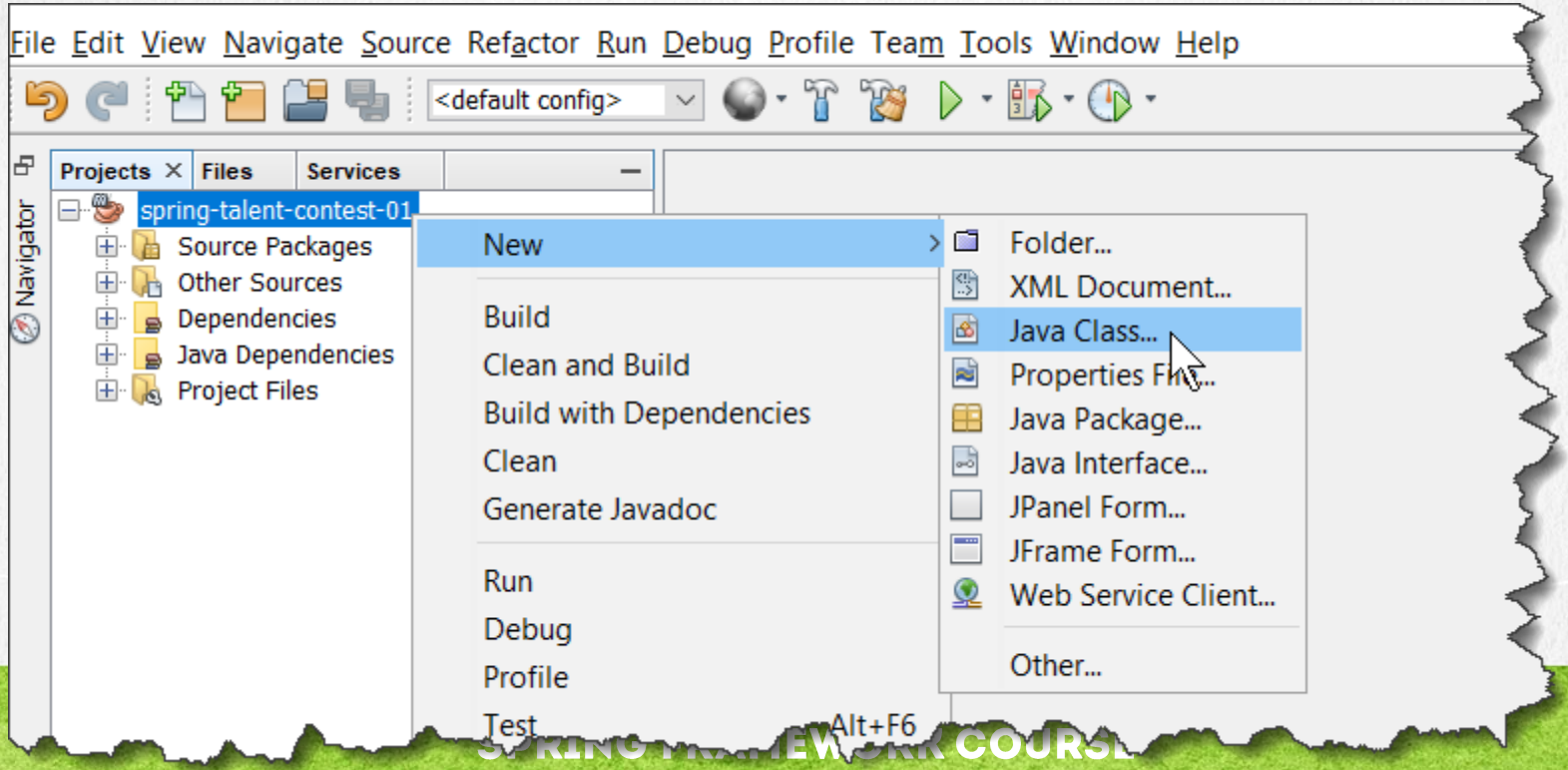
ExecutionException.java:

Click to download

```
package competitors;

class ExecutionException extends RuntimeException {

    public ExecutionException() {
    }

    public ExecutionException(String message) {
        super(message);
    }
}
```

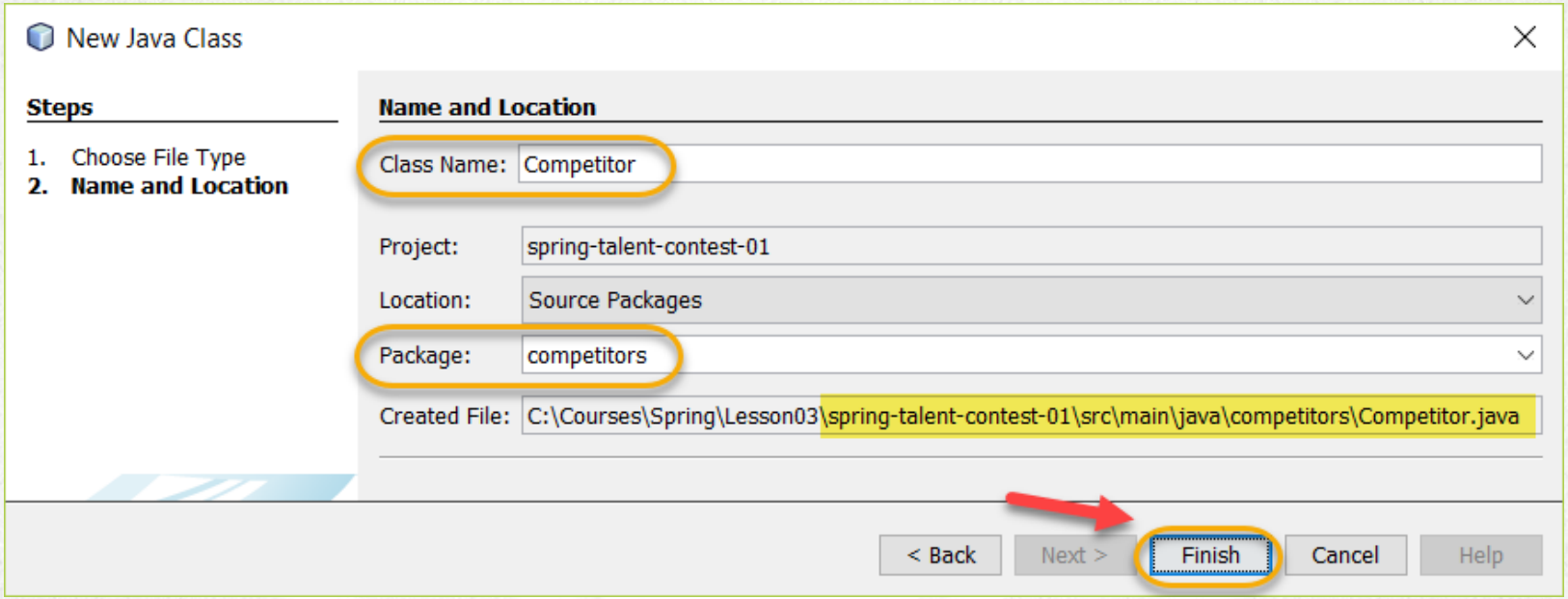
8. CREATE A JAVA CLASS

Create the Competitor.java interface:



8. CREATE A JAVA CLASS

Create the Competitor.java interface:



9. MODIFY THE FILE

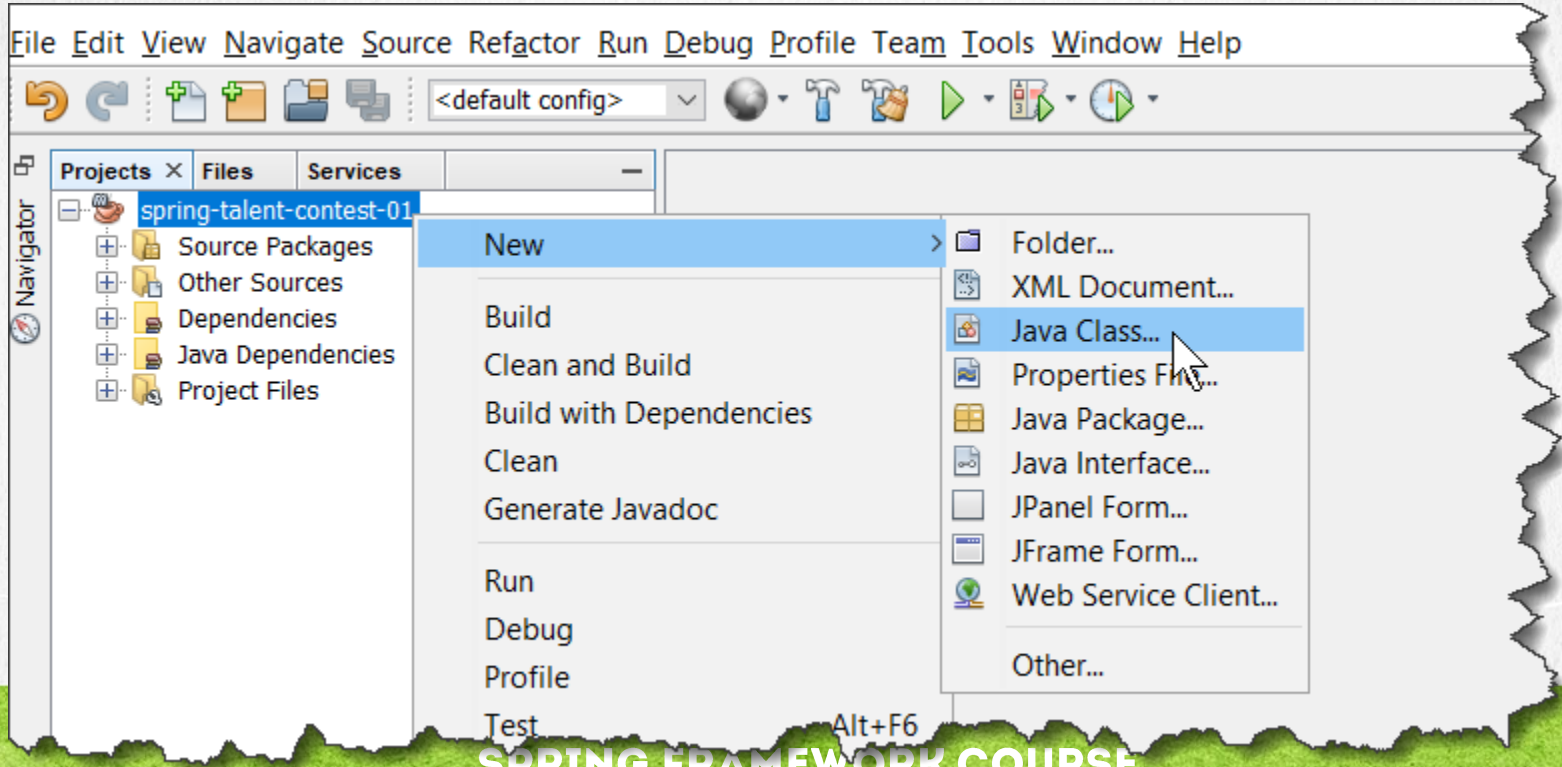
Competitor.java:

Click to download

```
package competitors;

public interface Competitor {

    public void execute() throws ExecutionException;
}
```

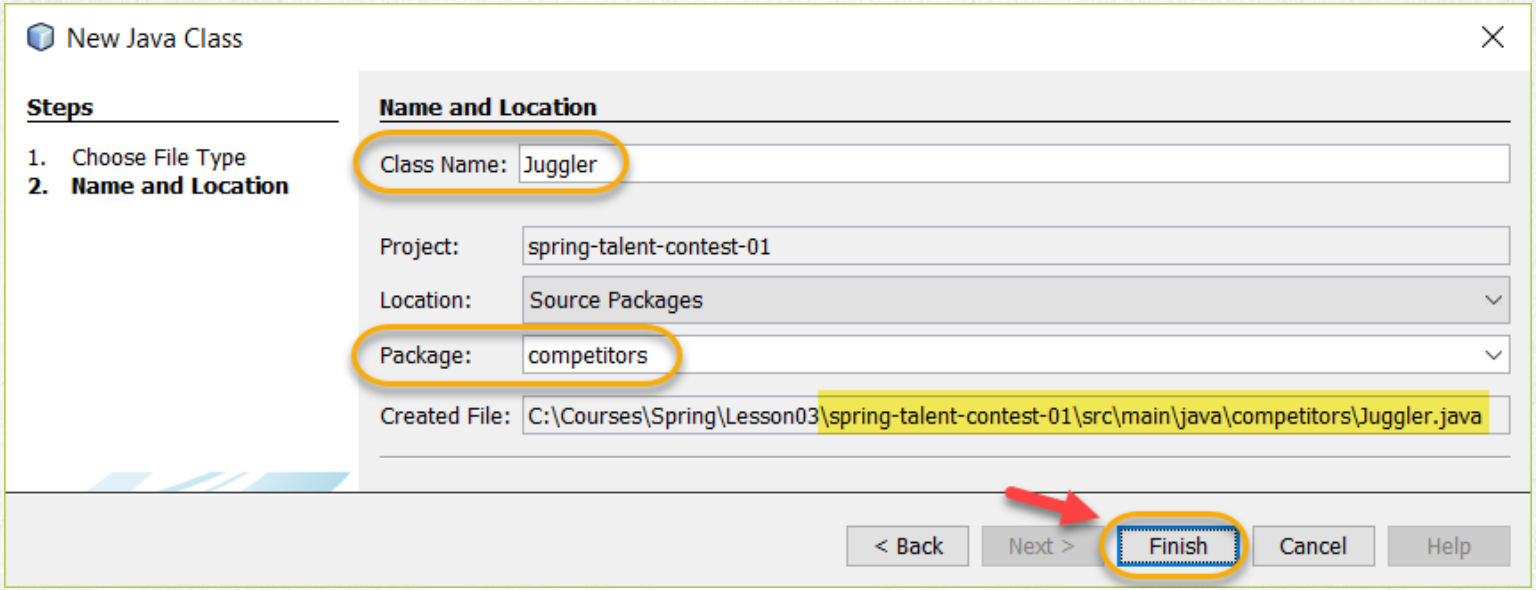
10. CREATE A JAVA CLASS

We create the Juggler.java class:



10. CREATE A JAVA CLASS

We create the Juggler.java class:



11. MODIFY THE FILE

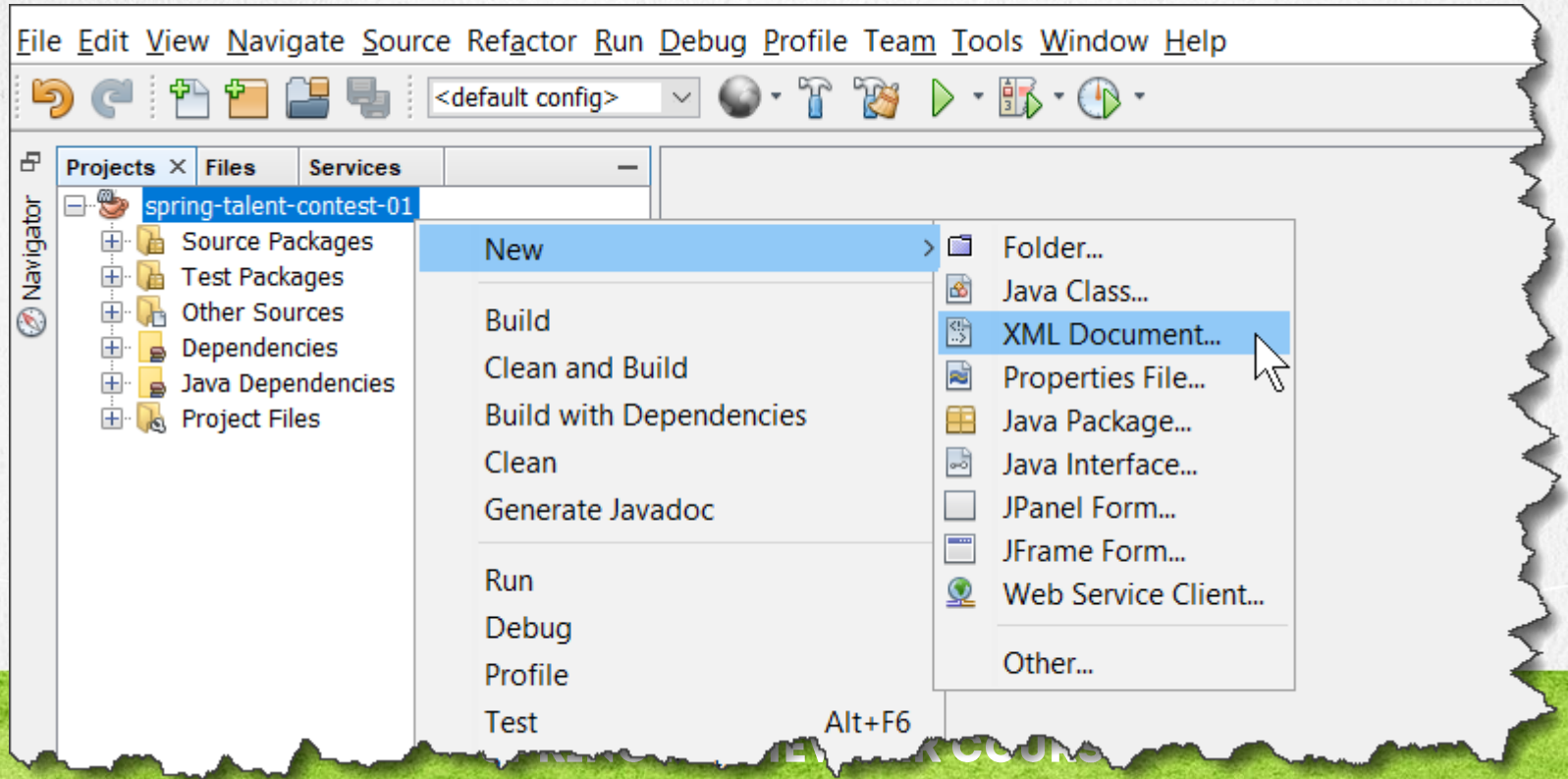
Juggler.java:

Click to download

```
package competitors;

public class Juggler implements Competitor {

    private int balls = 5;

    public Juggler() {
    }

    public Juggler(int balls) {
        this.balls = balls;
    }

    @Override
    public void execute() throws ExecutionException {
        System.out.println("juggling " + this.balls + " balls");
    }

    public int getBalls() {
        return balls;
    }

    public void setBalls(int balls) {
        this.balls = balls;
    }
}
```

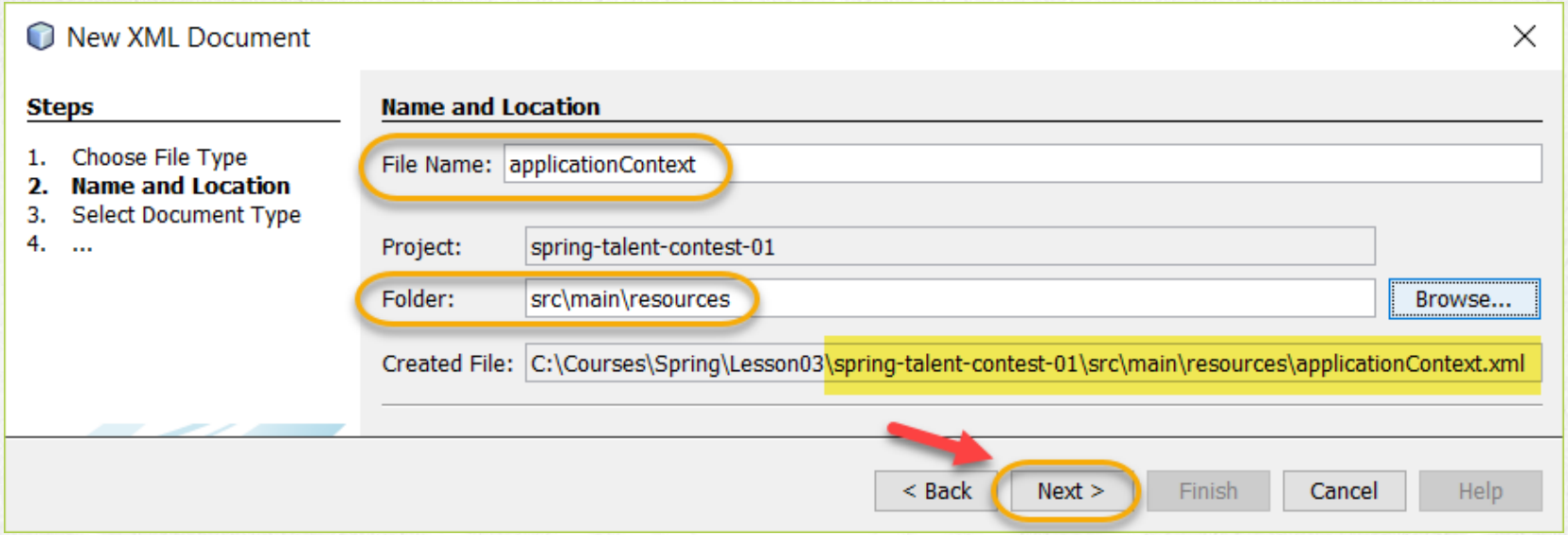
12. CREATE AN XML FILE

Create the applicationContext.xml file:



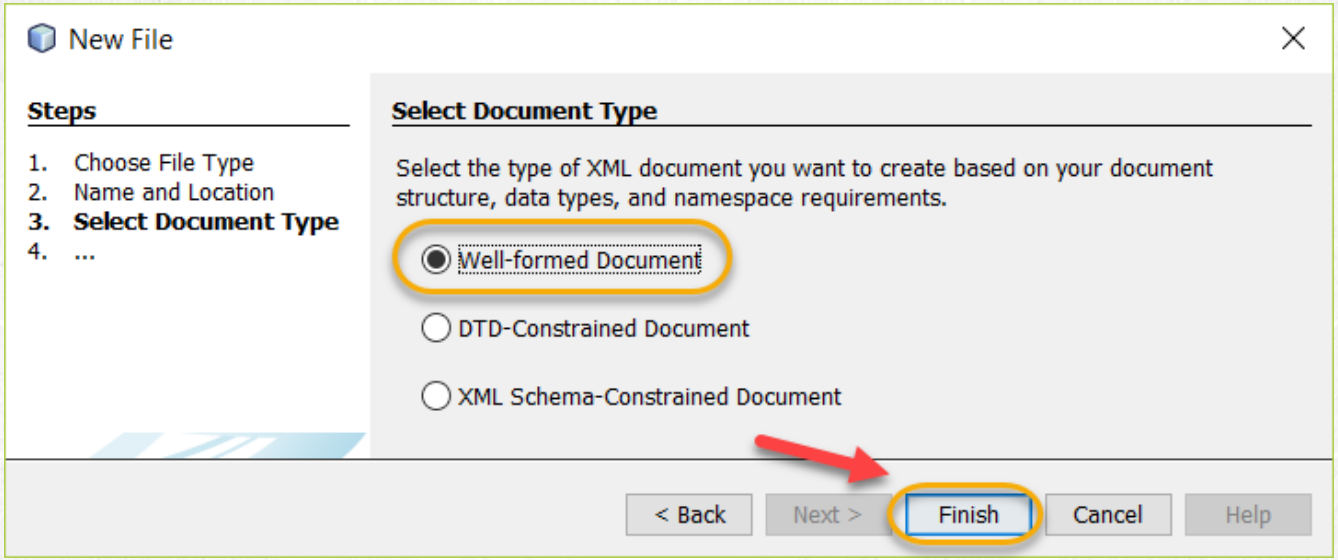
12. CREATE AN XML FILE

Create the applicationContext.xml file:



12. CREATE AN XML FILE

Create the applicationContext.xml file:



13. MODIFY THE FILE

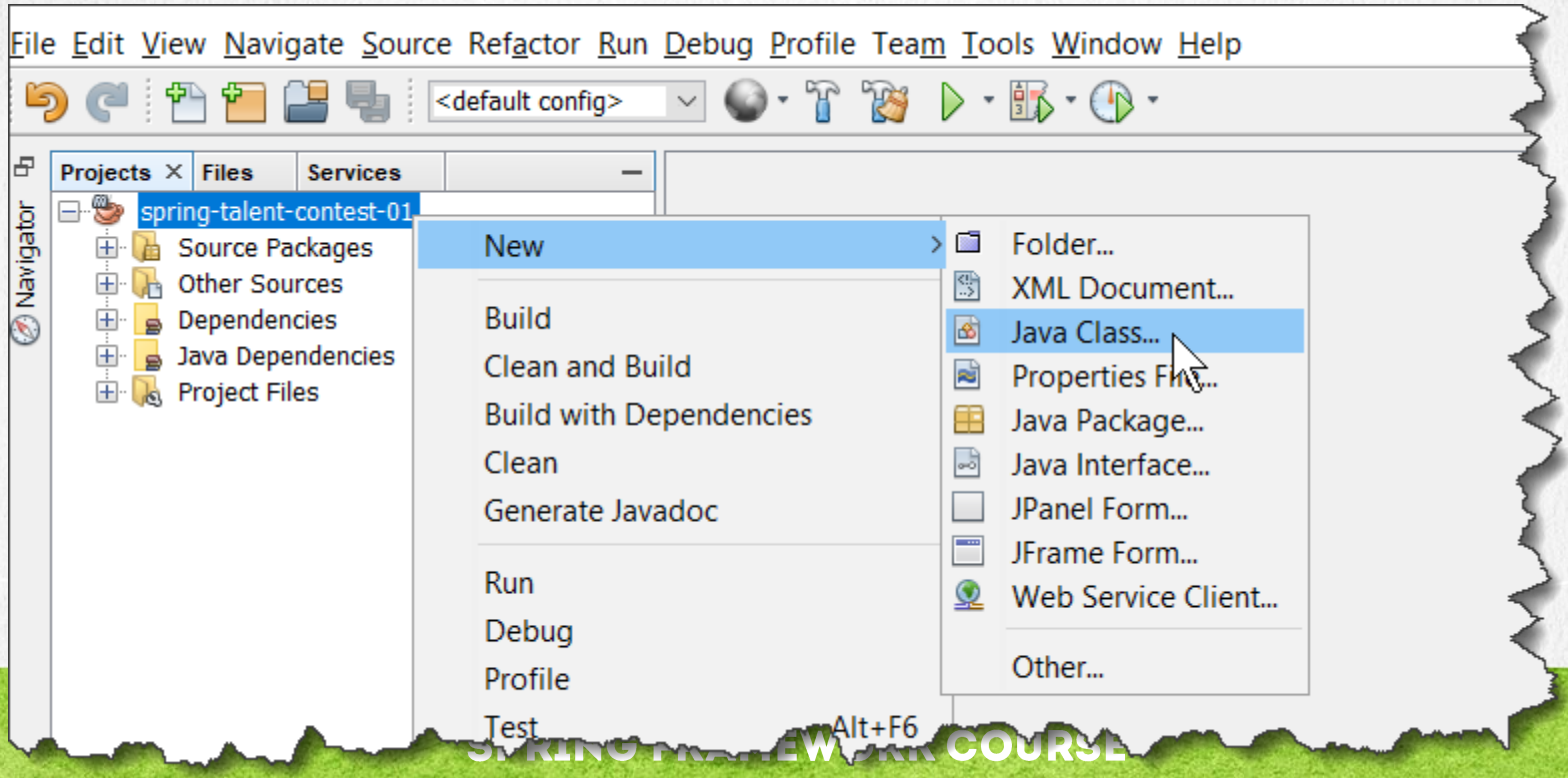
applicationContext.xml:

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">

    <!--  Equivalent code in Java:
    Juggler juggler = new Juggler(); -->
    <bean id="juggler" class="competitors.Juggler" />

</beans>
```

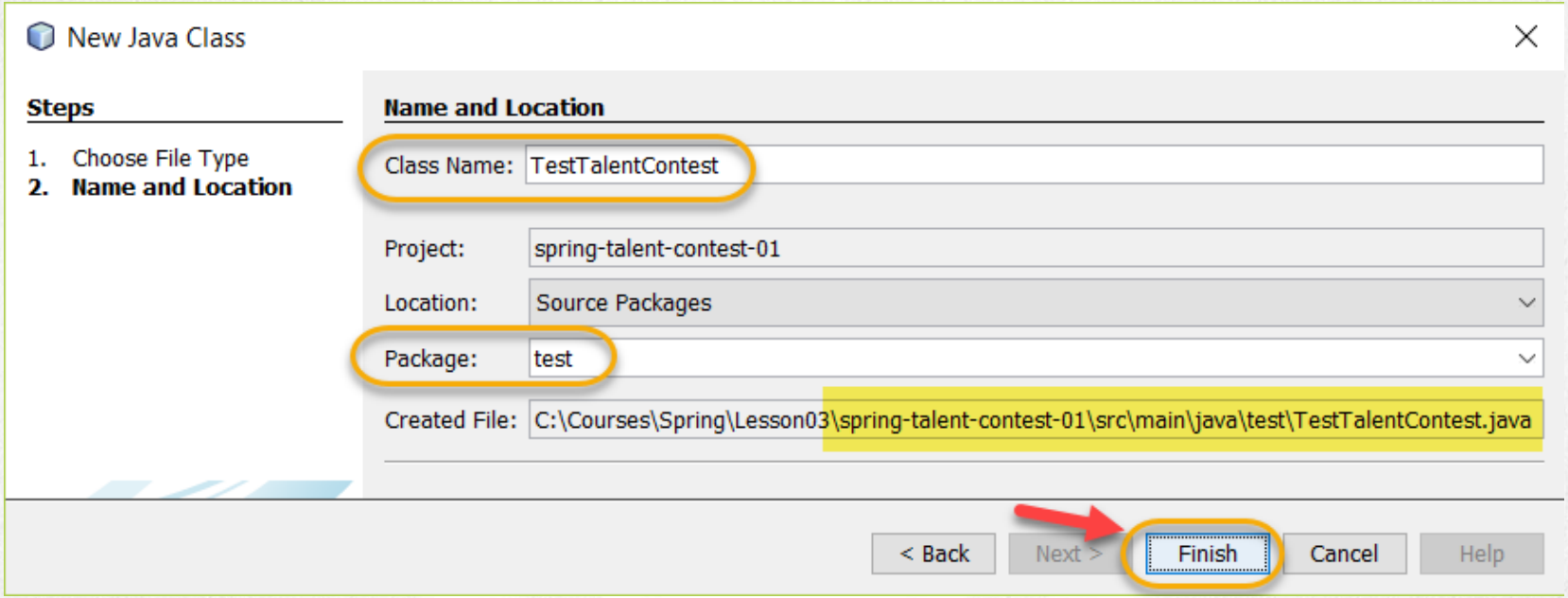
14. CREATE A NEW CLASS

We create the TestTalentContest .java class:



14. CREATE A NEW CLASS

We create the TestTalentContest .java class:



15. MODIFY THE CODE

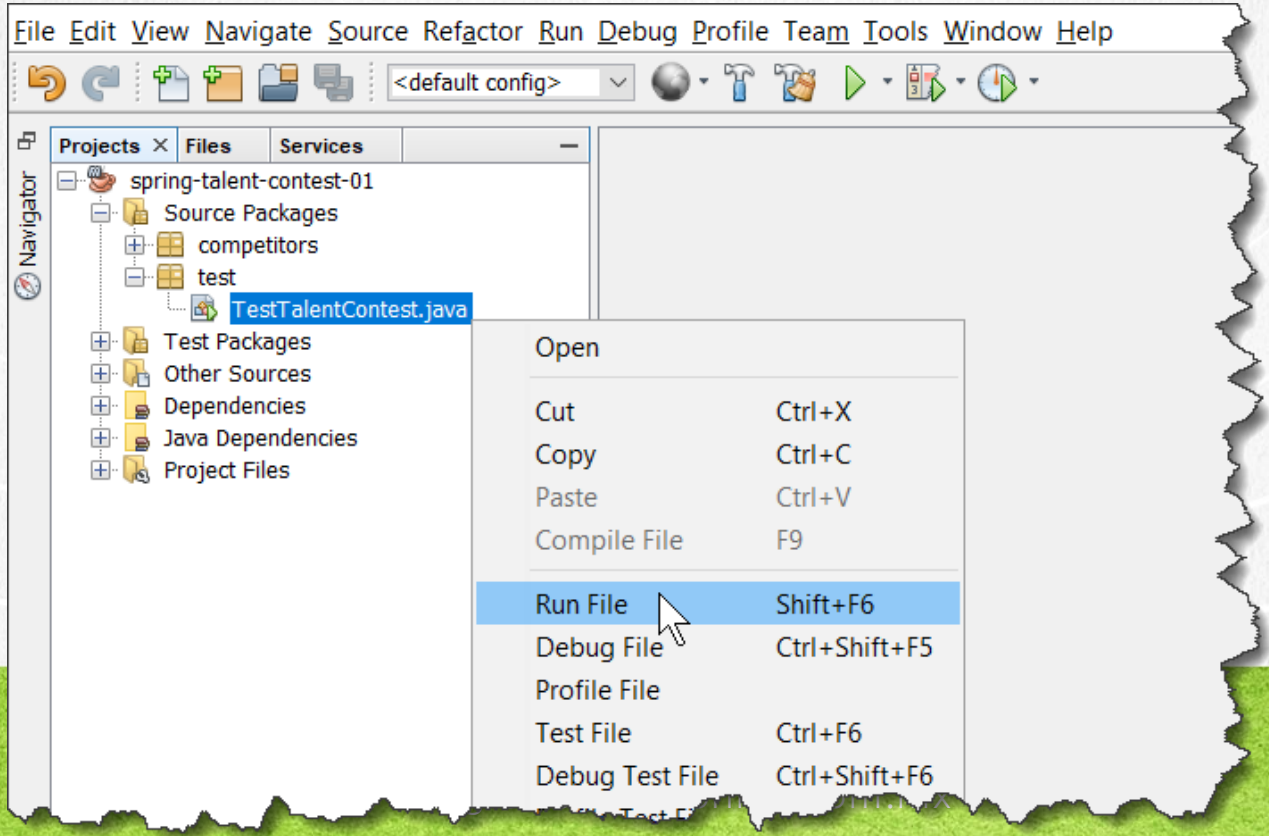
TestTalentContest.java:

Click to download

```
package test;

import competitors.Competitor;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestTalentContest {

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

        Competitor competitor1 = (Competitor) ctx.getBean("juggler");
        competitor1.execute();

    }
}
```

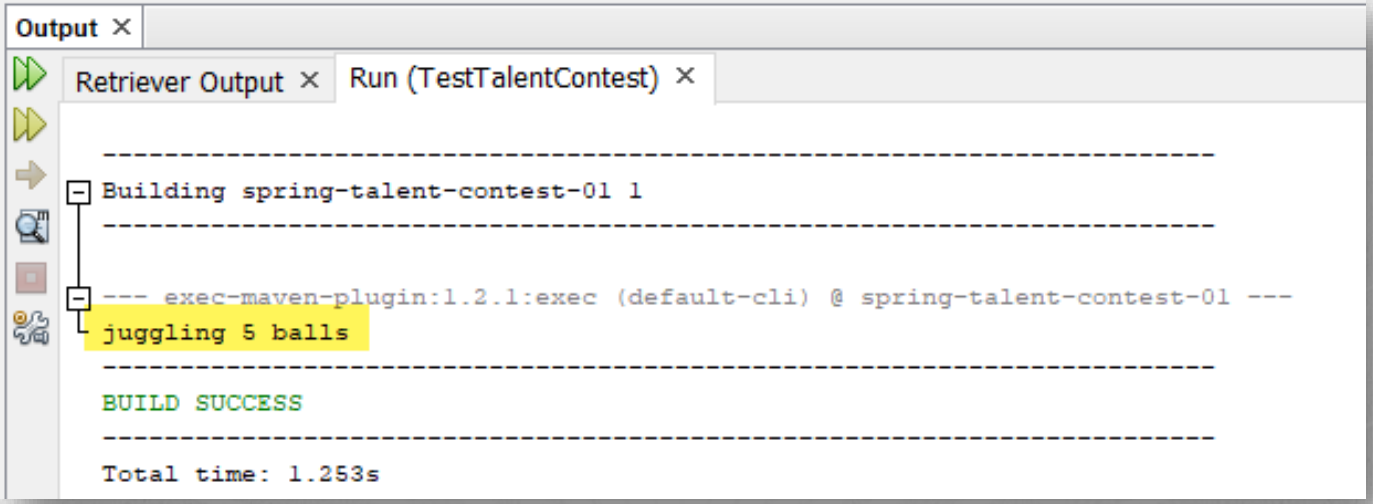
16. EXECUTE THE PROJECT

Execute the project:



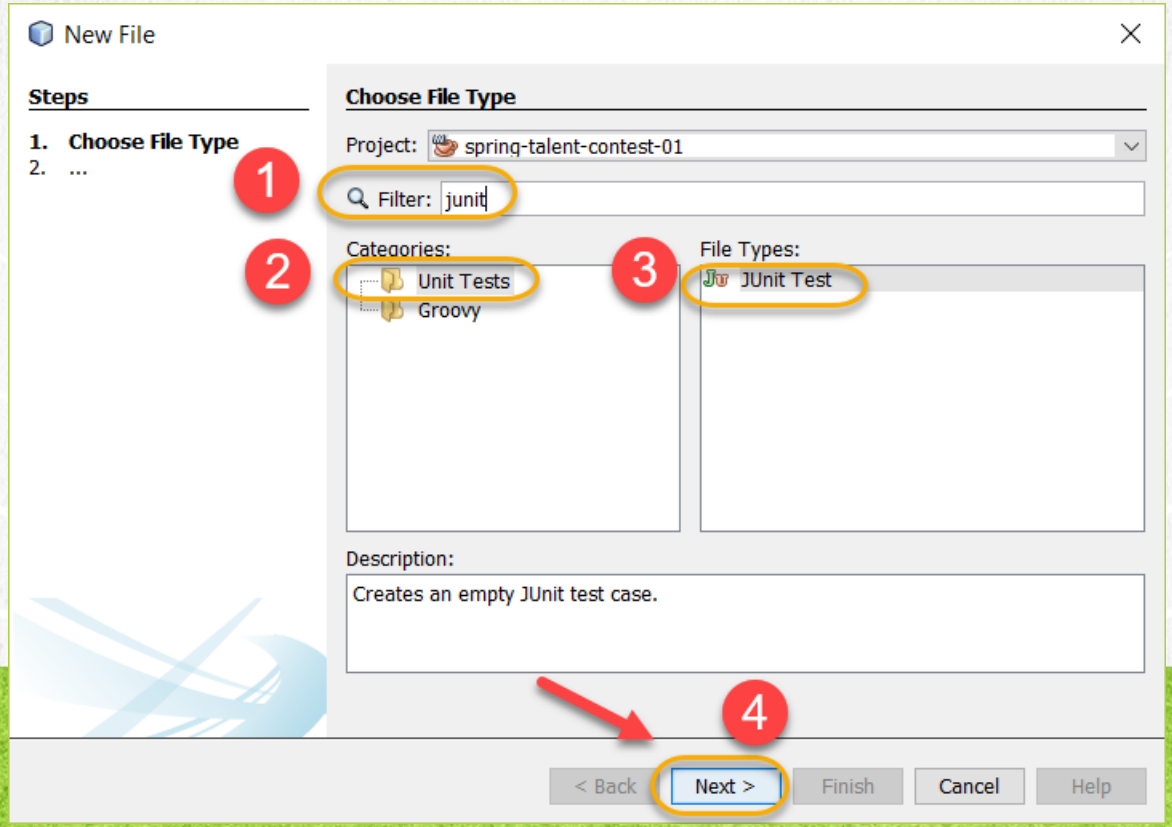
16. EXECUTE THE PROJECT

We execute the project. The result is as follows:



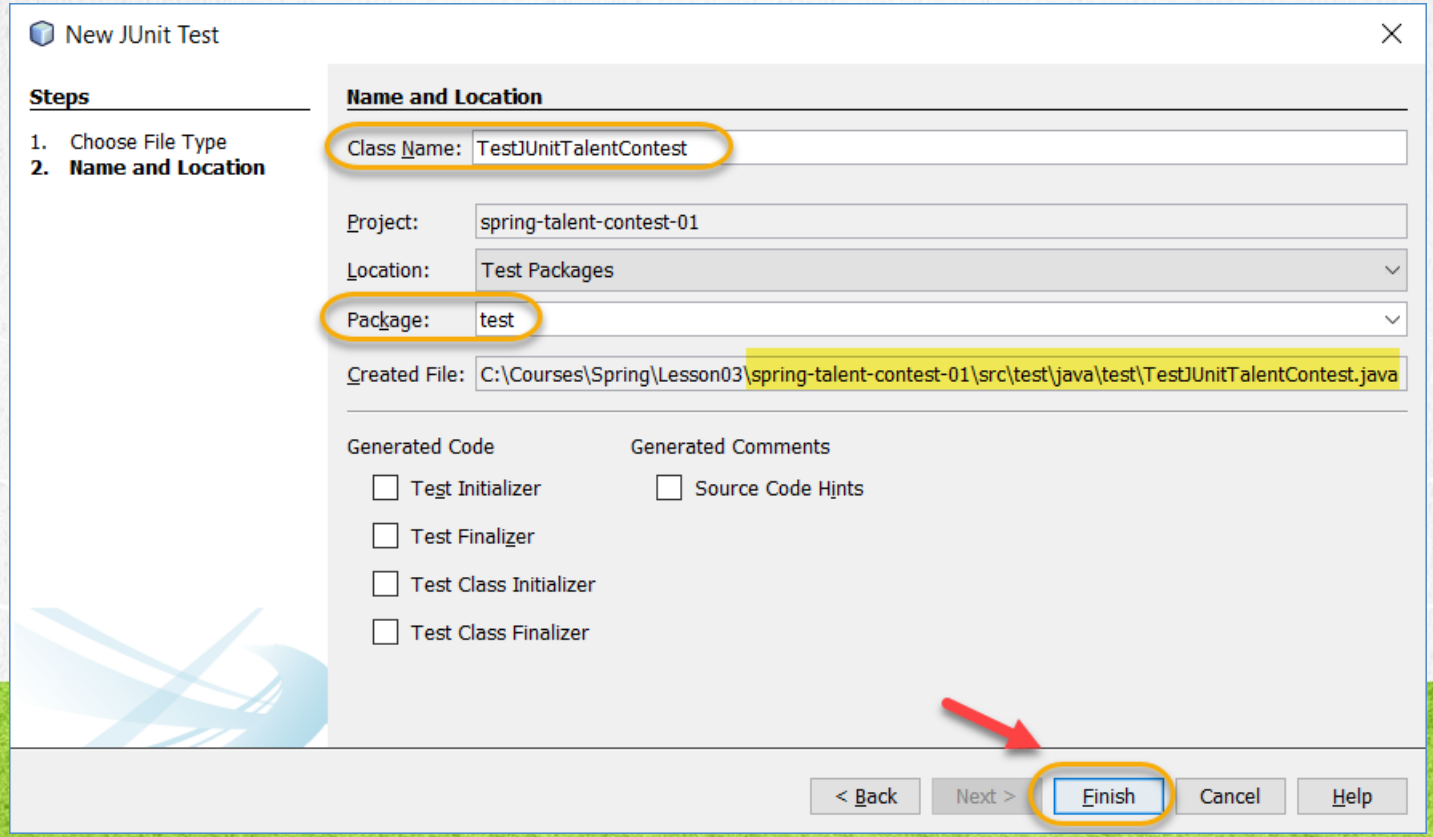
17. CREATE A TEST UNIT CLASS

Create the Junit TestTalentContest.java:



17. CREATE A TEST UNIT CLASS

Create the Junit TestTalentContest.java:



18. MODIFY THE FILE

TestJUnitTalentContest.java:

Click to download

```
package test;

import competitors.*;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.apache.logging.log4j.*;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TestJUnitTalentContest {

    Logger log = LogManager.getRootLogger();
    private Competitor competitor1;

    @BeforeEach
    public void before() {
        log.info("Starting Spring Framework");
        ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
        log.info("getting the first Competitor");
        competitor1 = (Competitor) ctx.getBean("juggler");
    }
```

18. MODIFY THE FILE

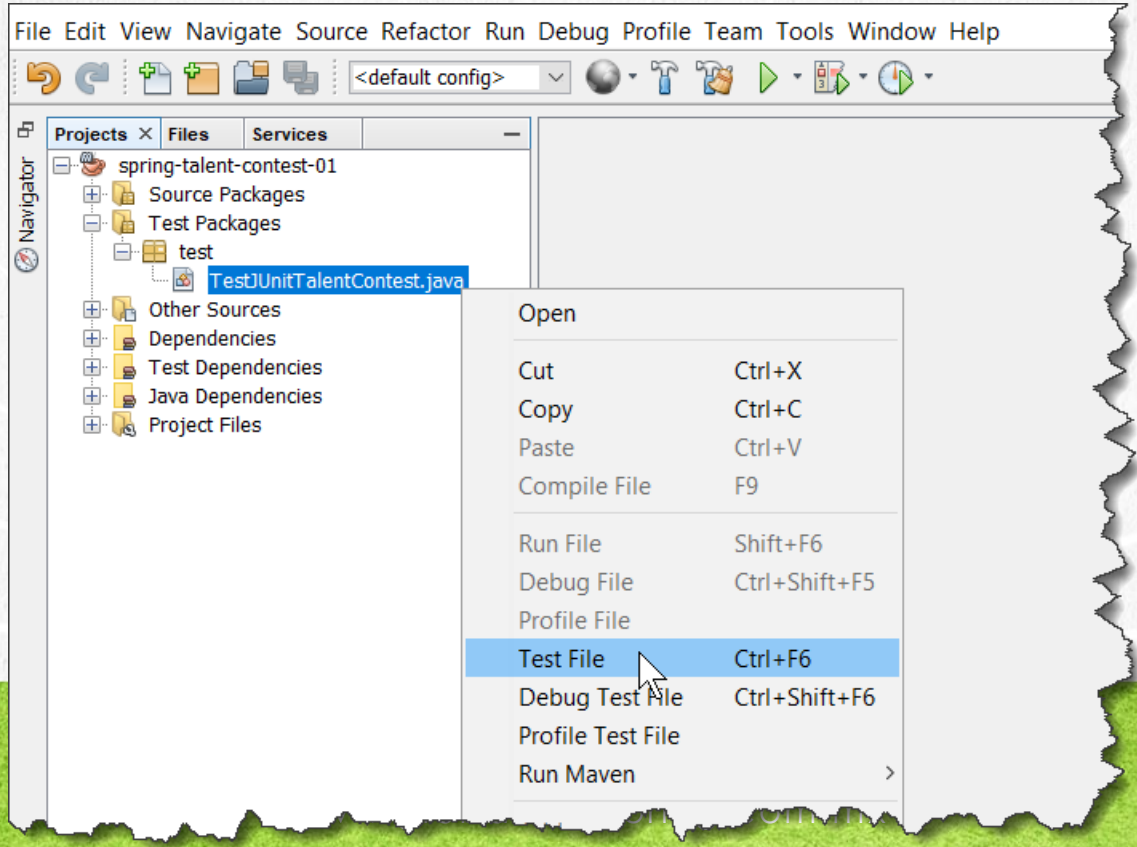
TestJUnitTalentContest.java:

Click to download

```
    @Test
    public void testJuggler() {
        log.info("Start executing Juggler");

        int ballsTest = 5;
        competitor1.execute();
        assertEquals(ballsTest, ((Juggler) competitor1).getBalls());

        log.info("Finish executing Juggler");
    }
}
```

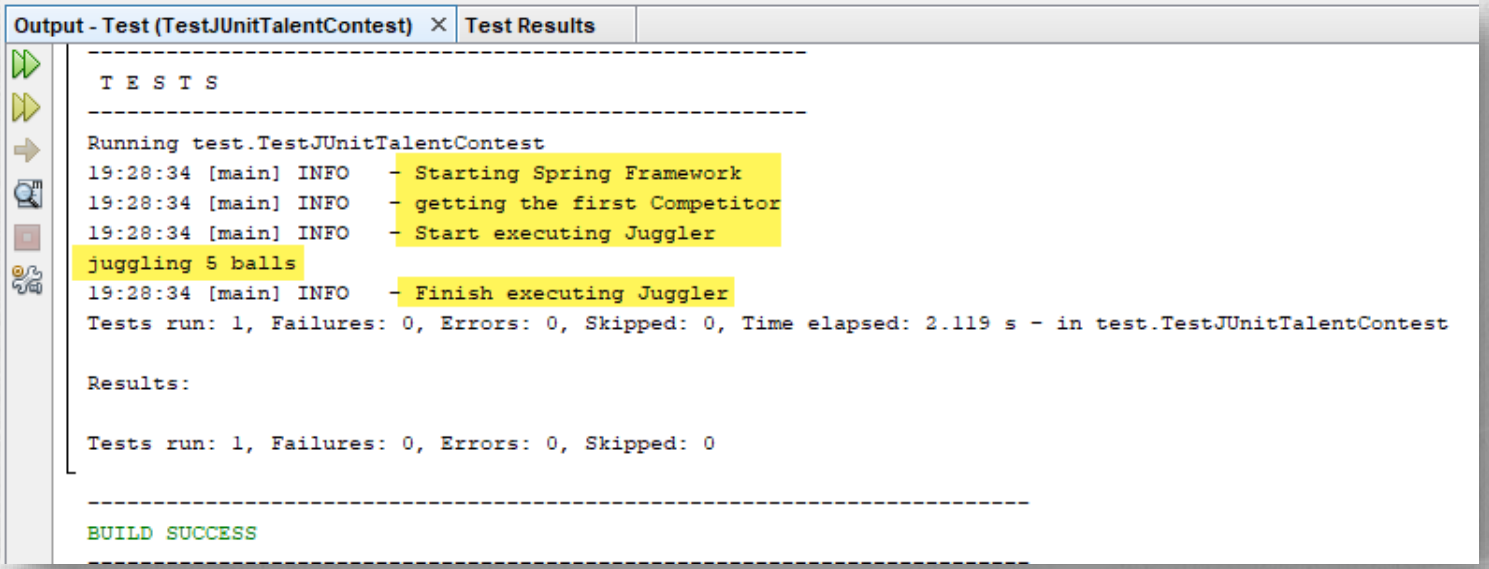
19. EXECUTE THE TEST

Execute the test:



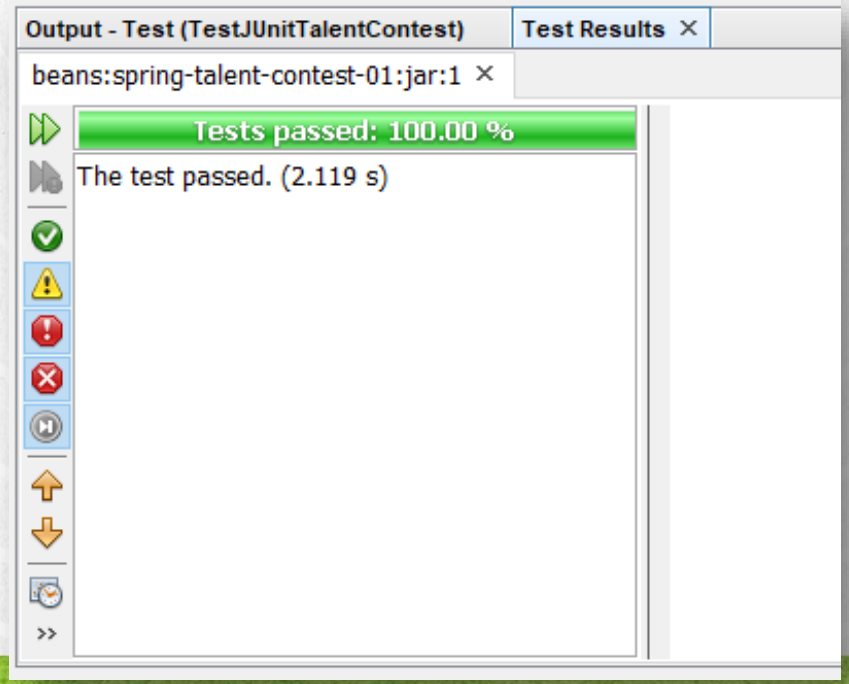
19. EXECUTE THE TEST

We execute the test. The result is as follows:



19. EXECUTE THE TEST

We execute the test. The result is as follows:



EXERCISE CONCLUSION

With this exercise we have laid the foundations for many of the exercises that we will create throughout this course.

The base project of the Talent Competition will be developed through several lessons to put into practice the concepts of Spring that we will study.

In this first version we have created the Competitor interface, and a first class called Juggler.java. Later we did a test with both a standard Java class, but we also did a unit test to check the use of the Spring factory and retrieved the Spring bean defined in the applicationContext.xml file.

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta