前提条件：

1、 安装了 Java 的 jdk

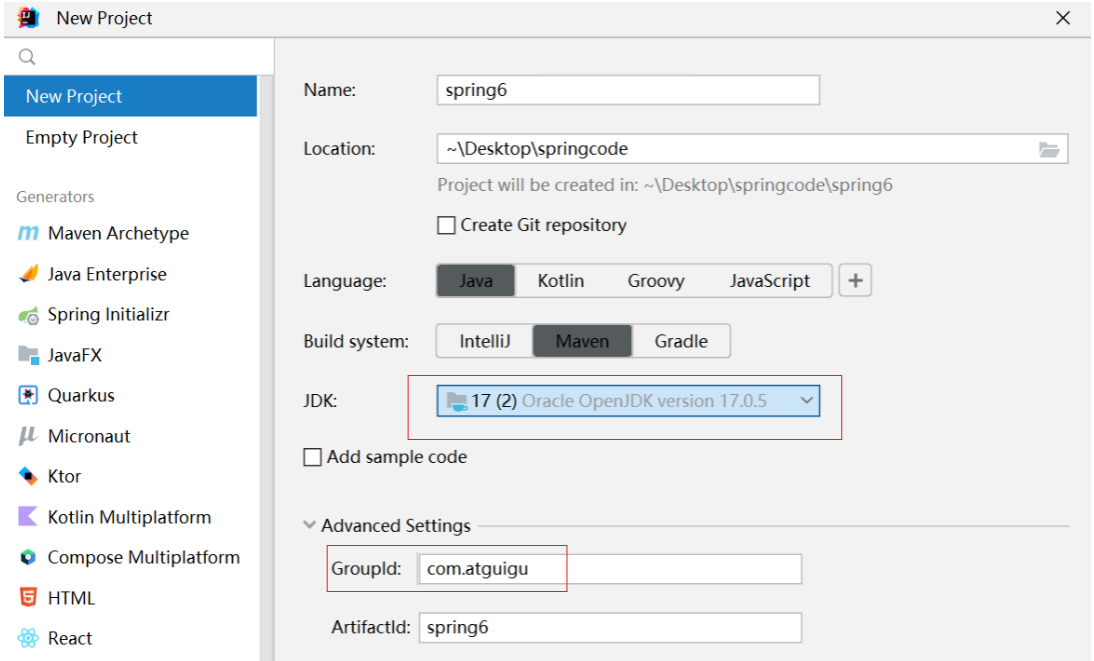
2、 安装好了 maven

3、 安装专业版 ideal

入门案例：

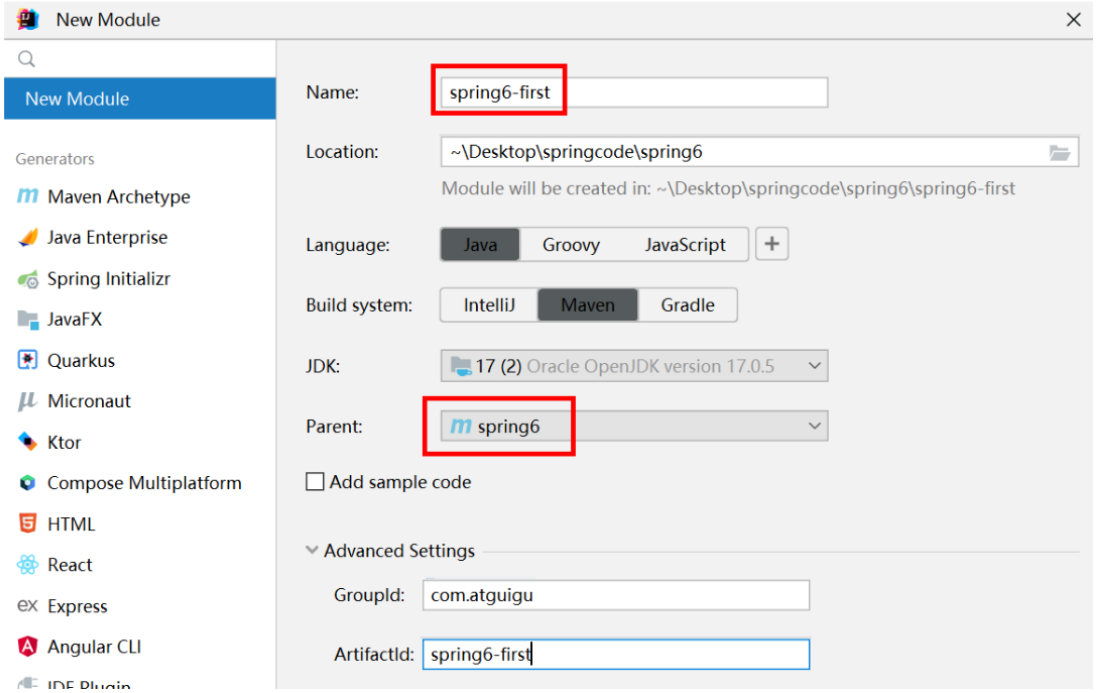
1、 新建父工程

在idea中，依次单击 File -> New -> Project -> New Project



2、 构建子模块

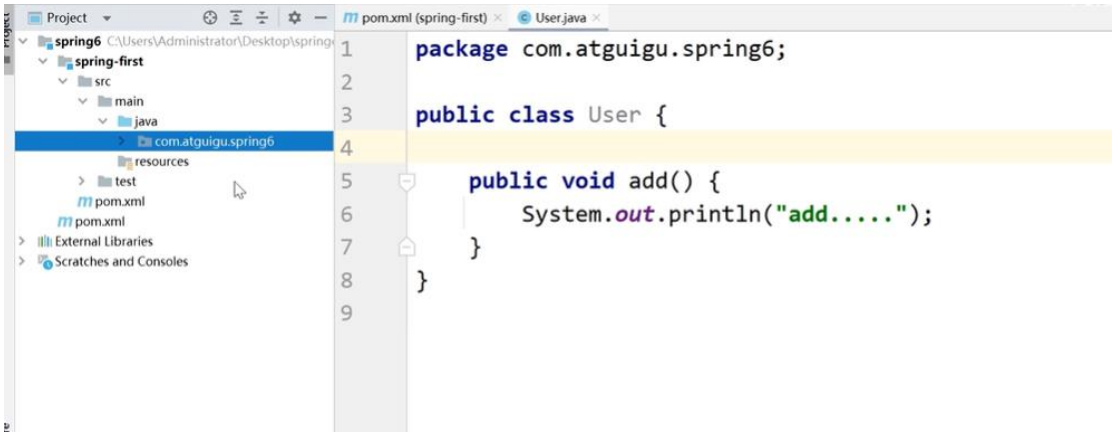
（2）构建子模块spring6-first



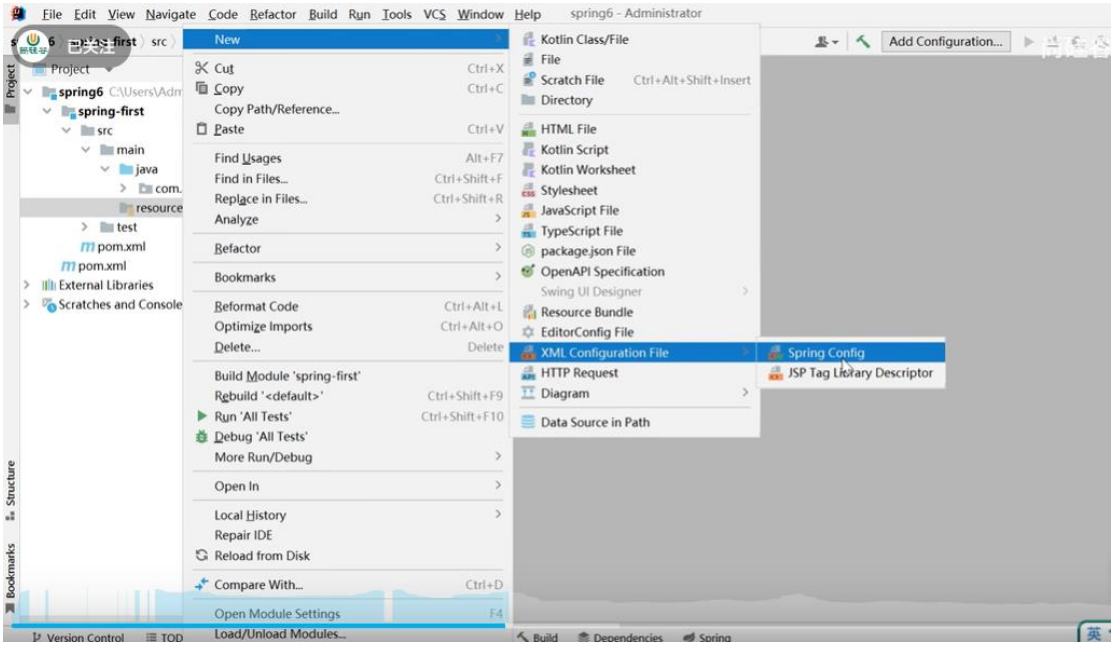
3、 在 spring-first 的 pom.xml 引入依赖

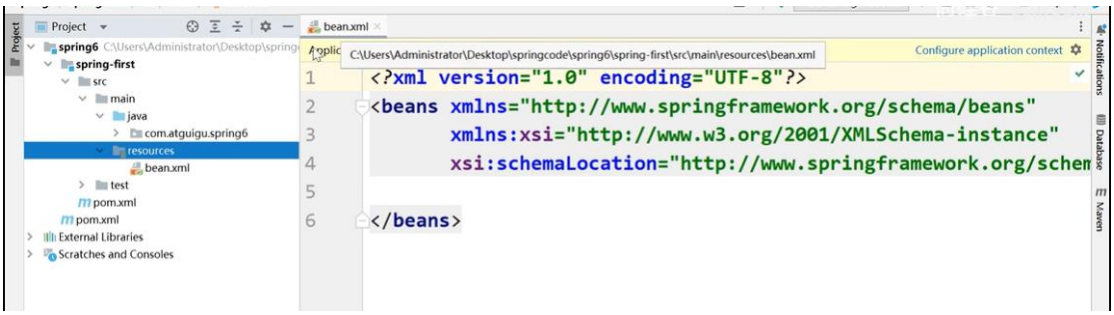
```
<dependencies>
    <!--spring context 依赖-->
    <!--当你引入 Spring Context 依赖之后，表示将 Spring 的基础依赖引入了-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>6.0.2</version>
    </dependency>

    <!--junit5 测试-->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.6.3</version>
    </dependency>
</dependencies>
```

4、 写入一个方法



5、 根据 spring 的规范建立配置文件





6、 在配置文件中创建对象

```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.spri

    <!--完成user对象创建
        bean标签
            id属性：唯一标识
            class属性：要创建对象所在类的全路径（包名称+类名称）
    -->
    <bean id="user" class="com.atguigu.spring6.User"></bean>
</beans>
```

对比

```java
    public static void main(String[] args) {
        User user = new User();
        user.add();
    }
}
```

7、 创建测试类

```java
package com.atguigu.spring6.bean;

import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```java
public class TestUser {

    @Test
    public void testUserObject() {
        //加载spring配置文件，对象创建
        ApplicationContext context =
                new ClassPathXmlApplicationContext( configLocation: "bean.xml");

        //获取创建的对象
        User user = (User)context.getBean( name: "user");
        System.out.println(user);

        //使用对象调用方法进行测试
        user.add();
    }
}
```

8、 完成第一次入门案例的实现

```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/schem

    <!--完成user对象创建
        bean标签
            id属性：唯一标识
            class属性：要创建对象所在类的全路径（包名称+类名称）
    -->
    <bean id="user" class="com.atguigu.spring6.User"></bean>
</beans>
```

```java
//获取创建的对象
User user = (User)context.getBean( name: "user");
System.out.println("1:"+user);

//使用对象调用方法进行测试
System.out.print("2:");
user.add();
}
```

新加的，用于区别

```
D:\java\jdk-17.0.5\bin\java.exe ...
1:com.atguigu.spring6.User@957e06
2:add.....

Process finished with exit code 0
```

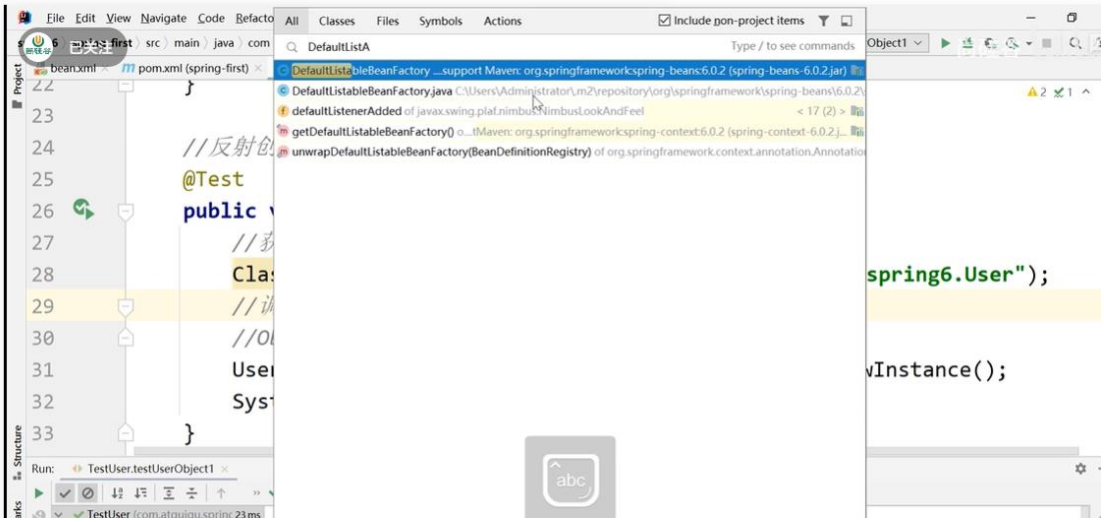
1、证明无参数构造的执行

没截图，就是写个无参构造

2、反射创建对象

4、 创建对象放在哪里?

查看源码:



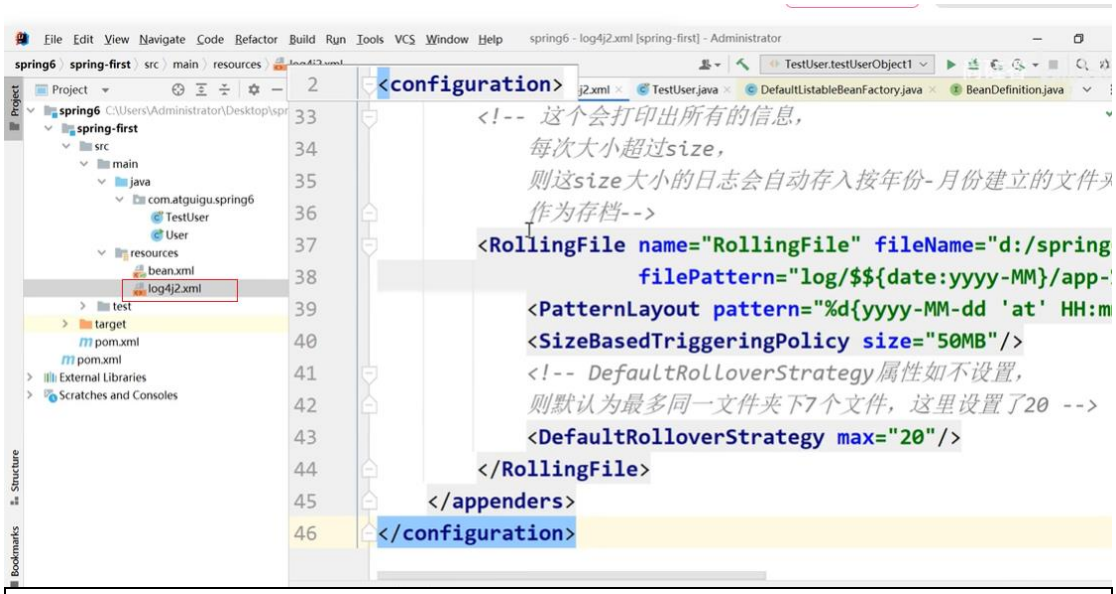
找到:

下面是整合 log4j2 的日志框架:

1、 引入依赖→spring-firsh –> pom.xml

```
    <!--log4j2 的依赖-->
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.19.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-slf4j2-impl</artifactId>
        <version>2.19.0</version>
</dependency>
```

2、 类路径下建立配置文件

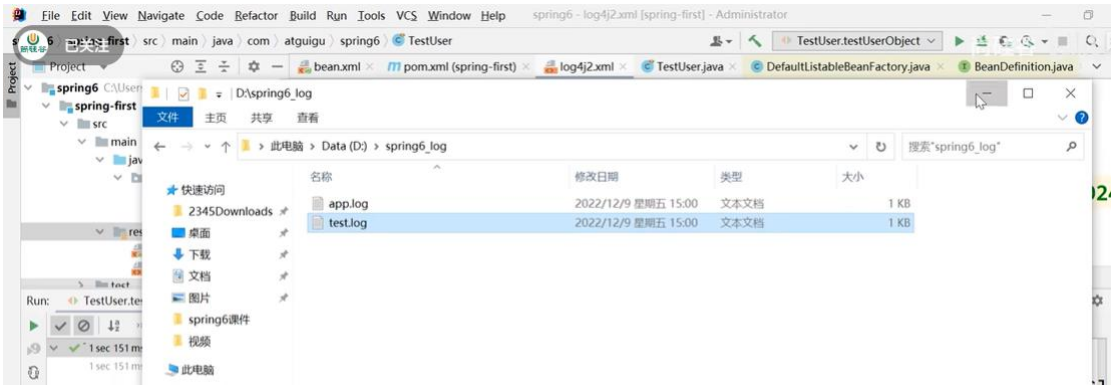



```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <loggers>
        <!--
            level 指定日志级别，从低到高的优先级:
                TRACE < DEBUG < INFO < WARN < ERROR < FATAL
```

```
                trace：追踪，是最低的日志级别，相当于追踪程序的执行
                debug：调试，一般在开发中，都将其设置为最低的日志级别
                info：信息，输出重要的信息，使用较多
                warn：警告，输出警告的信息
                error：错误，输出错误信息
                fatal：严重错误
        -->
        <root level="DEBUG">
            <appender-ref ref="spring6log"/>
            <appender-ref ref="RollingFile"/>
            <appender-ref ref="log"/>
        </root>
    </loggers>

    <appenders>
        <!--输出日志信息到控制台-->
        <console name="spring6log" target="SYSTEM_OUT">
            <!--控制日志输出的格式-->
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss SSS} [%t] %-3level %logger{1024} - %msg%n"/>
        </console>

        <!--文件会打印出所有信息，这个 log 每次运行程序会自动清空，由 append 属性决定，适合临时测试用-->
        <File name="log" fileName="d:/spring6_log/test.log" append="false">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"/>
        </File>

        <!-- 这个会打印出所有的信息，
            每次大小超过 size，
            则这 size 大小的日志会自动存入按年份-月份建立的文件夹下面并进行压缩，
            作为存档-->
        <RollingFile name="RollingFile" fileName="d:/spring6_log/app.log"
                     filePattern="log/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
            <PatternLayout pattern="%d{yyyy-MM-dd 'at' HH:mm:ss z} %-5level %class{36} %L %M - %msg%xEx%n"/>
            <SizeBasedTriggeringPolicy size="50MB"/>
            <!-- DefaultRolloverStrategy 属性如不设置，
            则默认为最多同一文件夹下 7 个文件，这里设置了 20 -->
            <DefaultRolloverStrategy max="20"/>
        </RollingFile>
```

```
    </appenders>
</configuration>
```

3、测试，执行并输出到配置文件



4、手动创建日志信息