# 前提条件:

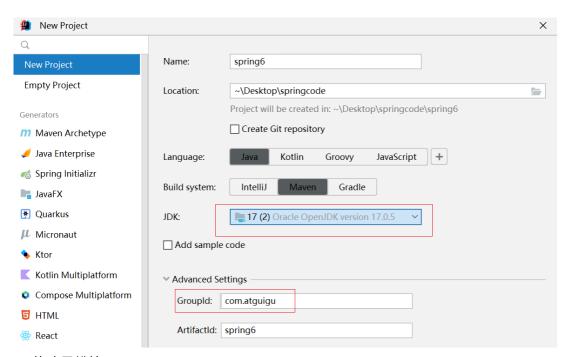
- 1、安装了 Java 的 jdk
- 2、安装好了 maven
- 3、安装专业版 ideal

# 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 New Module Name: spring6-first Location: ~\Desktop\springcode\spring6 Generators Module will be created in: ~\Desktop\springcode\spring6\spring6-first Maven Archetype 🥖 Java Enterprise JavaScript + Language: Groovy Spring Initialize Build system: IntelliJ Maven Gradle JavaFX Quarkus JDK: 📜 17 (2) Oracle OpenJDK version 17.0.5 **Micronaut** m spring6 Parent: 🔖 Ktor Compose Multiplatform Add sample code **THIML** ✓ Advanced Settings React Groupld: com.atguigu ex Express

3、在 spring-first 的 pom.xml 引入依赖

Angular CLI

IDE Dingin

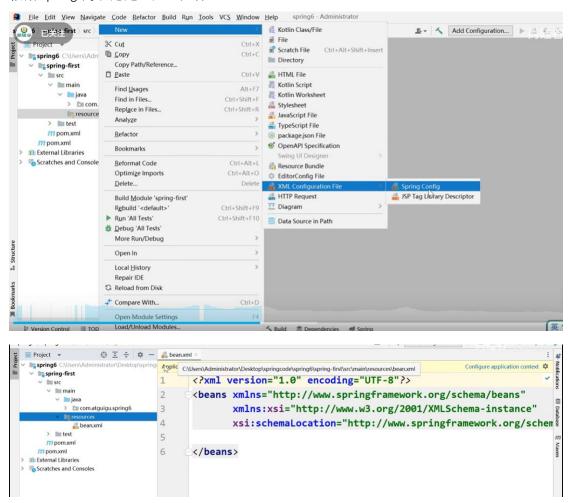
```
| <dependencies>
| <!--spring context 依赖-->
| <!--当你引入 Spring Context 依赖之后,表示将 Spring 的基础依赖引入了-->
| <dependency>
| <groupld>org.springframework</groupld>
| <artifactId>spring-context</artifactId>
| <version>6.0.2</version>
| </dependency>
| <groupld>org.junit.jupiter</groupld>
| <artifactId>junit-jupiter</groupld>
| <artifactId>junit-jupiter-api</artifactId>
| <version>5.6.3</version>
| </dependency>
| </dependency>
| </dependency>
| </dependencies>
```

ArtifactId: spring6-first

\_\_ 4、写入一个方法

```
⊕ 🚊 😤 💠 — 🎹 pom.xml (spring-first) × 💿 User.java ×
■ Project ▼
 spring6 C\Users\Administrator\Desktop\spring 1
                                           package com.atguigu.spring6;
   spring-first
      ∨ I main
                                   3
                                           public class User {
        ∨ ijava
                                  4
          m resources
      > 🛅 test
                                   5
                                                 public void add() {
      m pom.xml
                                                       System.out.println("add....");
                                   6
   m pom.xml
 Illi External Libraries
> Scratches and Consoles
                                   8
                                           }
                                   9
```

5、根据 spring 的规范建立配置文件



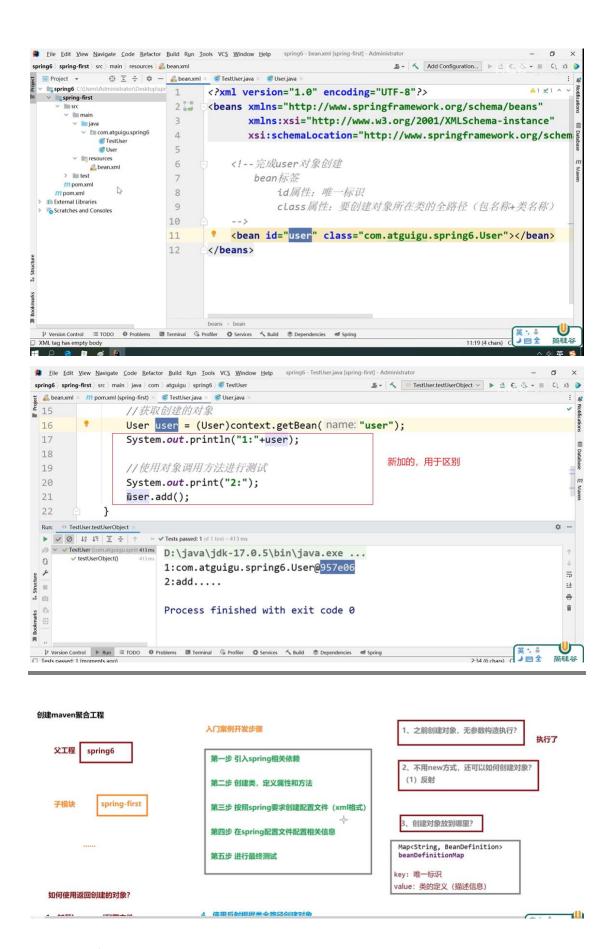
6、在配置文件中创建对象

```
bean.xml 💣 User.java ×
                                                                           A1 V1 ^ V
      <?xml version="1.0" encoding="UTF-8"?>
 2 deans xmlns="http://www.springframework.org/schema/beans"
 3
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4
            xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.spri
 5
         <!--完成user对象创建
 6
             bean标签
 7
                 id属性: 唯一标识
 8
                 class属性:要创建对象所在类的全路径(包名称+类名称)
 9
10
11
          <bean id="user" class="com.atguigu.spring6.User"></bean>
对比
            public static void main(String[] args) {
                  ₩ser user = new User();
                 user.add();
```

## 7、创建测试类

```
package com.atguigu.spring6.bean;
import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
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();
}
  III TODO ● Problems III Terminal ← Profiler ← Services ← Build ● Dependencies ● Spring
```

8、完成第一次入门案例的实现



#### 1、证明无参数构造的执行

## 没截图, 就是写个无参构造

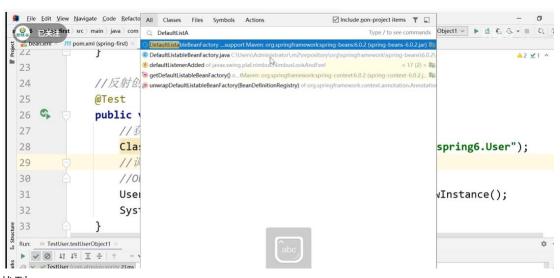
## 2、反射创建对象

```
如何使用返回创建的对象?
                                       4、使用反射根据类全路径创建对象
  1、加载bean.xml配置文件
                                      Class clazz = Class.forName("com.atguigu.spring6.User");
  2、对xml文件进行解析操作
  3、获取xml文件bean标签属性值
                                       //Object o = clazz.newInstance();
                                      User user = (User)clazz.getDeclaredConstructor().newInstance();
  id属性值和class属性值
-first ⟩ src ⟩ main ⟩ java ⟩ com ⟩ atguigu ⟩ spring6 ⟩ 💣 TestUser ⟩ 💣 🛊 testUserObject1
                                                       B → 🔨 

TestUser.testUserObject1 ∨ ▶ ± € € € → ■ ○ □
m pom.xml (spring-first) ×
               TestUser.iava
           System.out.print( 2: );
           user.add();
      }
      //反射创建对象
      @Test
      public void testUserObject1() throws Exception {
           //获取类CLass对象
           Class clazz = Class.forName( className: "com.atguigu.spring6.User");
           //调用方法创建对象
           //Object o = clazz.newInstance();
           User user = (User)clazz.getDeclaredConstructor().newInstance();
           System.out.println(user);
      }
  }
                                                                                            -Ille
```

#### 4、创建对象放在哪里?

#### 查看源码:



找到:

```
| Map of bean definition objects, keyed by bean name. | Map from bean name to merged BeanDefinitionHolder. | Map from Bean name to merged
```

# 下面是整合 log4i2 的日志框架:

1、引入依赖→spring-firsh -> pom.xml

2、 类路径下建立配置文件

```
<u># File Edit View Navigate Code Refactor Build Run Iools VCS Window Help</u> spring6 - log4j2.xml [spring-first] - Administrator
                                                              ♣ - 🔨 • TestUser.testUserObject1 v 🕨 🛎 € € + 🔳 C( a)
Configuration> j2.xml × @ TestUser.java × @ DefaultListableBeanFactory.java × @ BeanDefinition.java × :
             ⊕ <u>₹</u> <del>*</del> 2
  ■ Project *
  spring6 C:\Users\Administrator\Desktop\spr 33
                                          <!-- 这个会打印出所有的信息,
    spring-first
                                               每次大小超过size,
                         34
                         35
                                               则这size大小的日志会自动存入按年份-月份建立的文件系

∨ Im com.atguigu.spring6

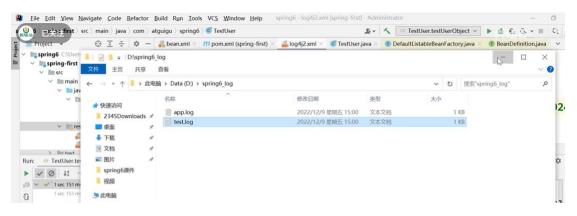
© TestUser

                                               作为存档-->
                         36
            © User
                         37
                                          <RollingFile name="RollingFile" fileName="d:/spring
                                                         filePattern="log/$${date:yyyy-MM}/app-
                                               <PatternLayout pattern="%d{yyyy-MM-dd 'at' HH:m
                         39
    > lest
    m pom.xml
                         40
                                               <SizeBasedTriggeringPolicy size="50MB"/>
                                               <!-- DefaultRolloverStrategy属性如不设置,
                         41
  III External Libraries
  Scratches and Consoles
                         42
                                               则默认为最多同一文件夹下7个文件,这里设置了20 -->
                         43
                                               <DefaultRolloverStrategy max="20"/>
                         44
                                          </RollingFile>
                         45
                                      </appenders>
                                 </configuration>
<?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>
       <!--輸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"</pre>
                  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

```
Elle Edit View Navigate Code Refactor Build Run Iools VCS Window Help spring6 - TestUser.java [spring-first] - Administrator
                                                                         ♣ - 🔨 • TestUser.testUserObject ∨ 🕨 👙 🧲 🗞 🕶 🗏
ıg6 ⟩ spring-first ⟩ src ⟩ main ⟩ java ⟩ com ⟩ atguigu ⟩ spring6 ⟩ © TestUser ⟩ € logger
lj bean.xml × 🞢 pom.xml (spring-first) × 🔒 log4j2.xml × 🎳 TestUser.java × 🌘 DefaultListableBeanFactory.java × 🕦 BeanDefinition.java × 🕲 User.java ×
3
       import org.junit.jupiter.api.Test;
4
       import org.slf4j.Logger;
5
       import org.slf4j.LoggerFactory; I
6
       import org.springframework.context.ApplicationContext;
7
       import org.springframework.context.support.ClassPathXmlApplicationContext;
8
       1 usage
9 6
       public class TestUser {
0
1
             //创建Logger对象
        private Logger logger = LoggerFactory.getLogger(TestUser.class);
2
3
Л
             @Toct
📱 Eile Edit View Navigate Code Refactor Build Run Iools VCS Window Help spring6 - TestUser.java [spring-first] - Administrator
pring6 | spring-first | src | main | java | com | atguigu | spring6 | © TestUser | @ = testUserObject
                                                                              ♣ - 🔨 • TestUser.testUserObject ∨ 🕨 💆 €
 🚜 bean.xml 🗴 🎢 pom.xml (spring-first) 🗴 🚜 log4j2.xml 🗴 💰 TestUser.java 🗡 ® DefaultListableBeanFactory.java 🗴 🔞 BeanDefinition.java 🗴 🍪 User.java
19
                      //获取创建的对象
 20
 21
                      User user = (User)context.getBean( name: "user");
                     System.out.println("1:"+user);
 22
 23
                      //使用对象调用方法进行测试
 24
 25
                      System.out.print("2:");
                      user.add();
 26
 27
                      //手动写日志
 28
                      logger.info("###执行调用成功了..");
 29
 30
 31
```