

# 实验1 Spring开发环境搭建

## (一) 实验目的

1. 通过本实验学会搭建Spring项目
2. 通过本实验掌握Spring的IoC功能

## (二) 实验内容

1. 新建Web工程
2. 创建applicationContext.xml 文件，引入Spring的约束
3. 将Spring的核心jar包文件导入项目
4. 分层实现service层和dao层bean
5. 在applicationContext.xml 配置每一层bean
6. 访问service层的bean对象

## (三) 实验要求

1. 实验过程要求每个同学根据实验指导书认真完成
2. 每个同学认真书写实验总结，不能有雷同

## (四) 实验步骤

### 1. 在IDEA中创建一个Maven项目

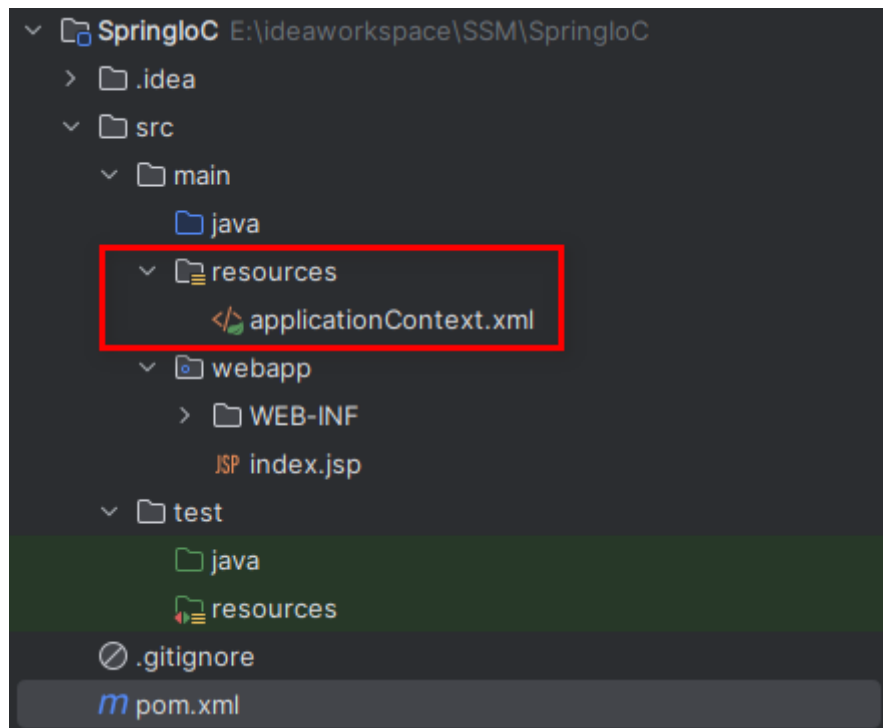
此步骤略，请参考 IDEA新建Maven项目实验。

### 2. 在项目pom.xml文件中引用Spring的核心jar包

The screenshot shows the IntelliJ IDEA interface. On the left, the project structure is visible, with the 'pom.xml' file highlighted in the 'test' directory. A red box around 'pom.xml' has a red arrow pointing to it with the text '双击打开'. On the right, the 'pom.xml' file is open, showing the Maven configuration. A red box highlights the 'spring-context' dependency, with a red arrow pointing to it and the text 'spring核心容器环境依赖包只需添加一个spring-context即可，4个核心包不需要全部添加'. The pom.xml content is as follows:

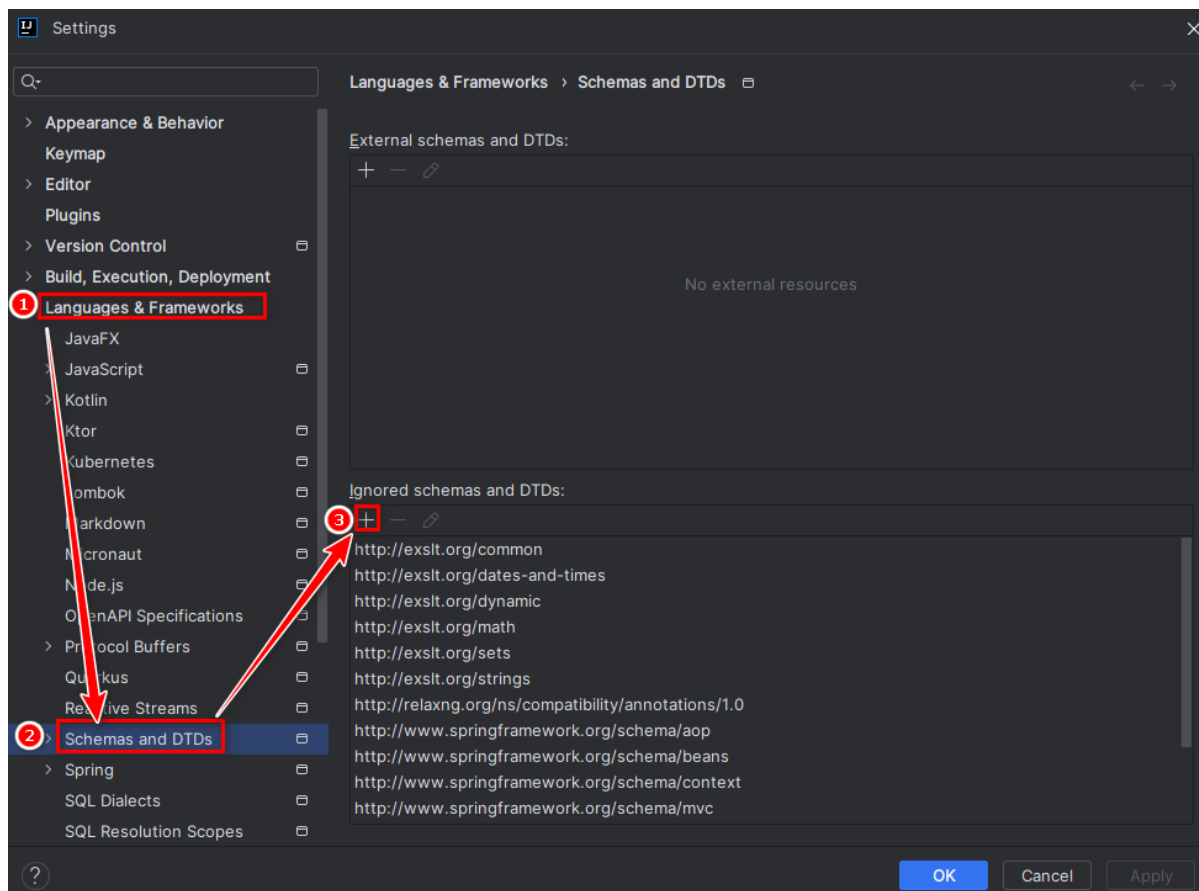
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>SpringIoC</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>SpringIoC Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>4.3.5.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>SpringIoC</finalName>
  </build>
</project>
```

### 3. 将applicationContext.xml文件引入工程src->main->resources目录下

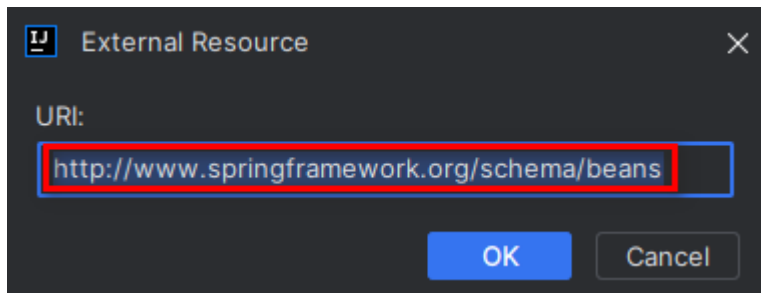


**注：**在拷贝applicationContext.xml文件过程中，如果applicationContext.xml文件报错，执行以下操作。

点击File->Setting，在弹出的对话框中执行：

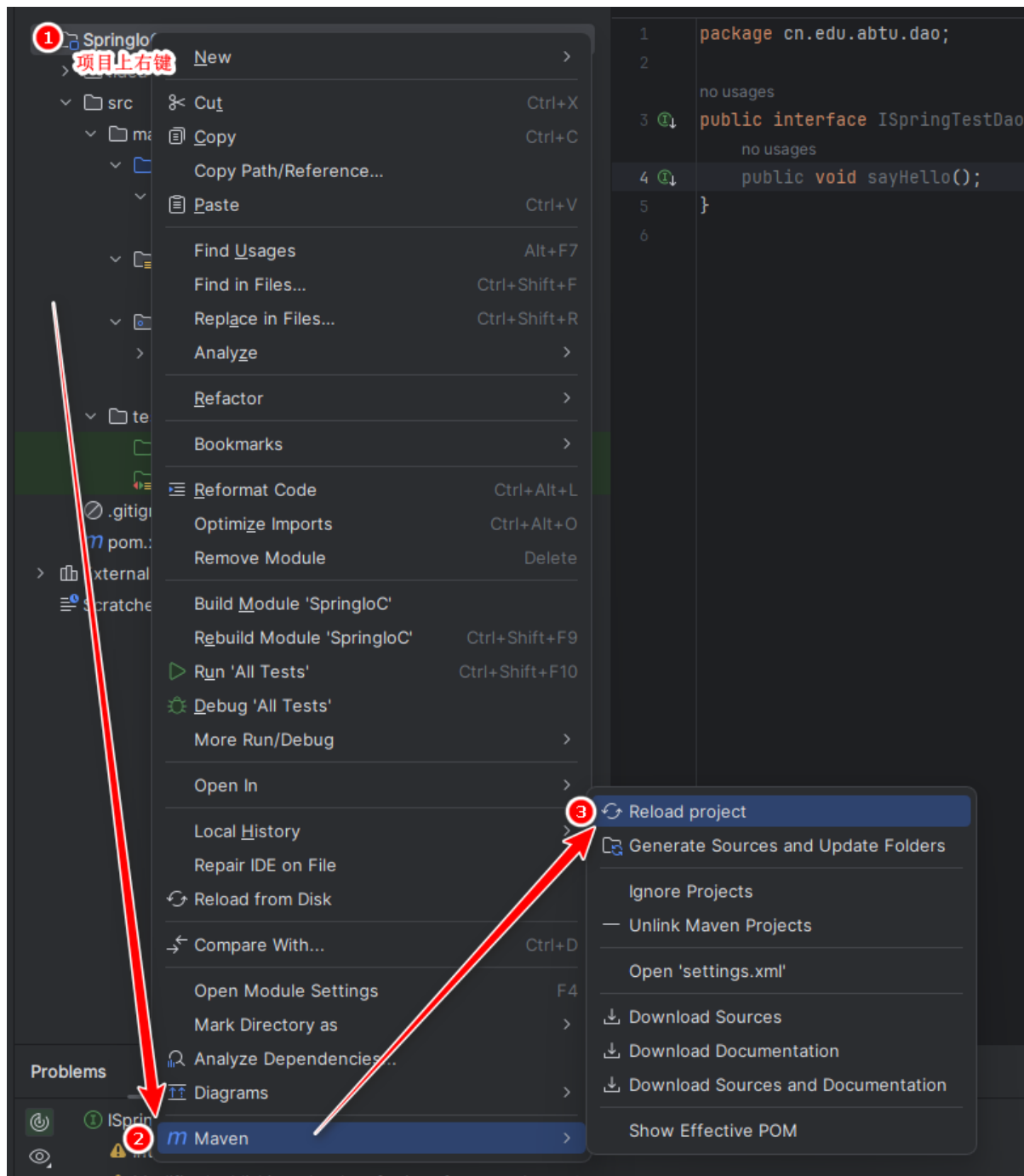


点击 **+** 后在弹出的对话框中填入



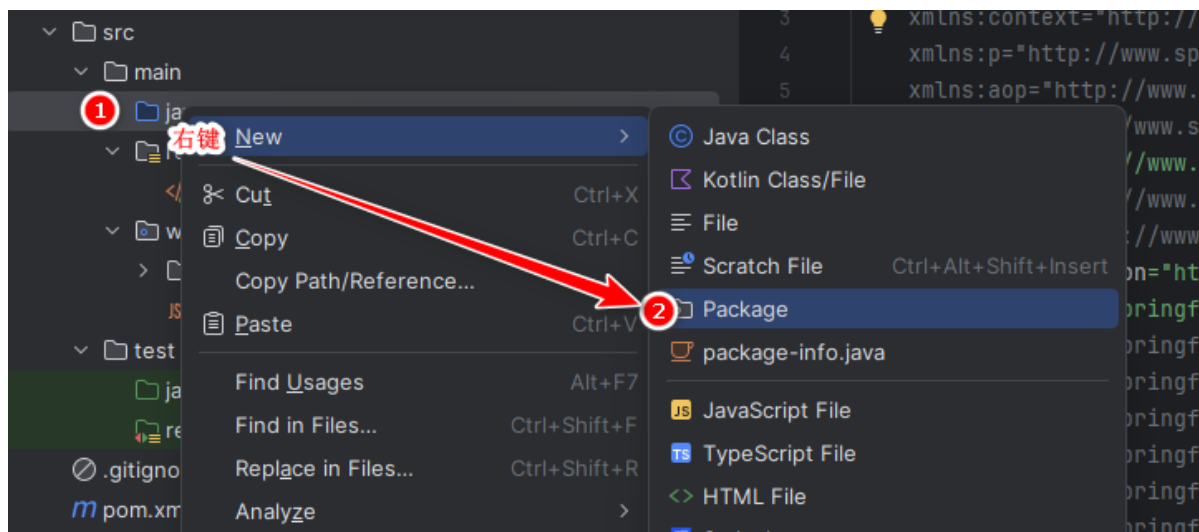
**注：** <http://www.springframework.org/schema/beans>为报错的行，哪一行报错，就把该行复制过来，添加忽略即可。**逐行操作。**

最后进行如下操作：

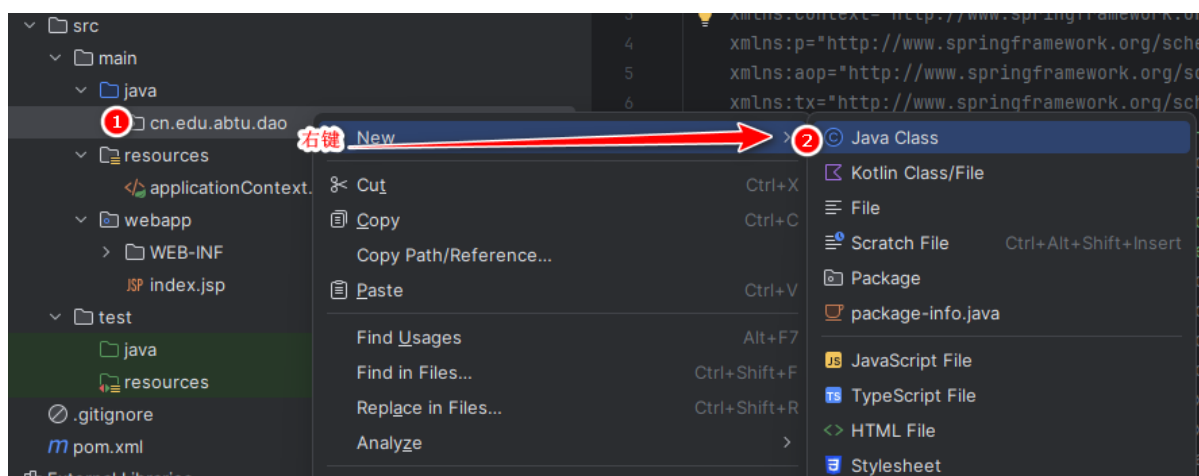


重新加载以下Maven项目即可。

#### 4. 在src->main->java下建立cn.edu.abtu.dao包



#### 5. 在cn.edu.abtu.dao包下建ISpringTestDao接口



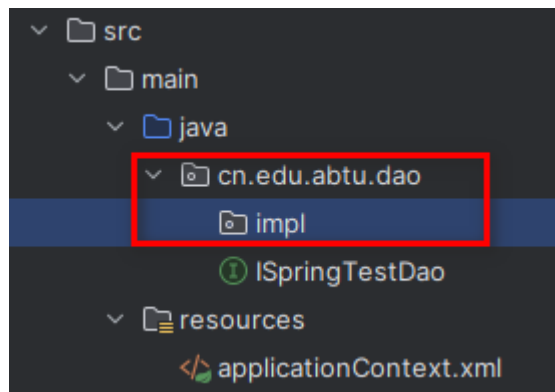
在弹出的对话框中输入接口名称ISpringTestDao



#### 6. 在ISpringTestDao接口中定义sayHello()方法

```
public interface ISpringTestDao {  
    no usages  
    public void sayHello();  
}
```

7. 在src->main->java下新建cn.edu.abtu.dao.impl包



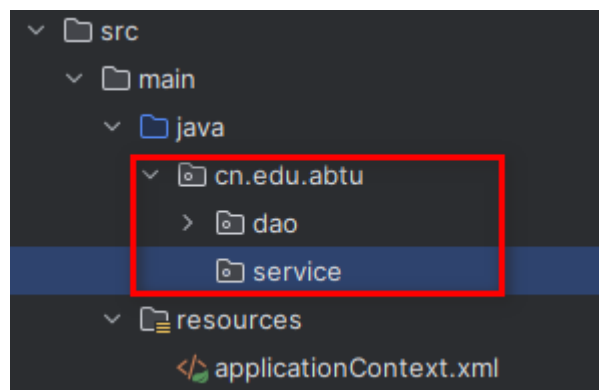
8. 在cn.edu.abtu.dao.impl下新建ISpringTestDao的实现类SpringTestDaoImpl, 并实现sayHello()方法

```
package cn.edu.abtu.dao.impl;

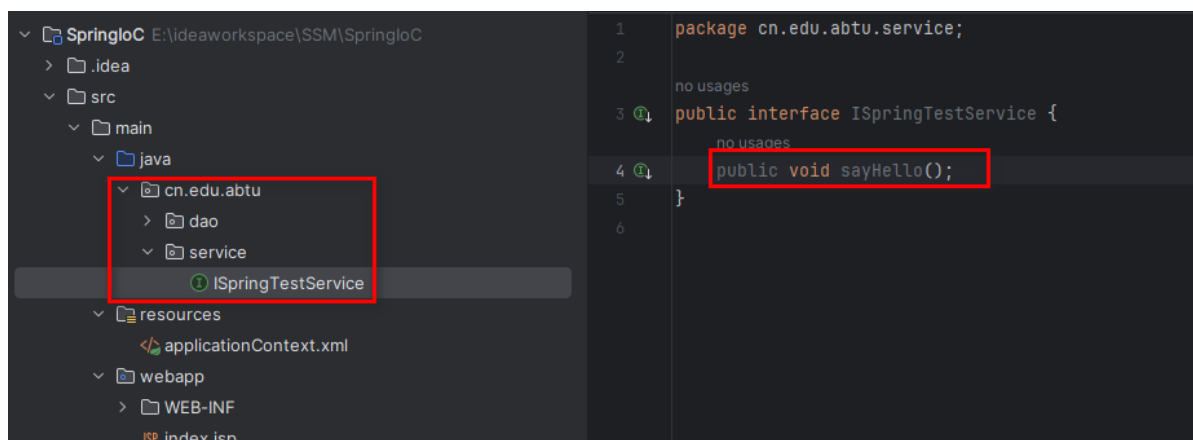
import cn.edu.abtu.dao.ISpringTestDao;

no usages
public class SpringTestDaoImpl implements ISpringTestDao {
    no usages
    public void sayHello() {
        System.out.println("Hello Spring!!!!");
    }
}
```

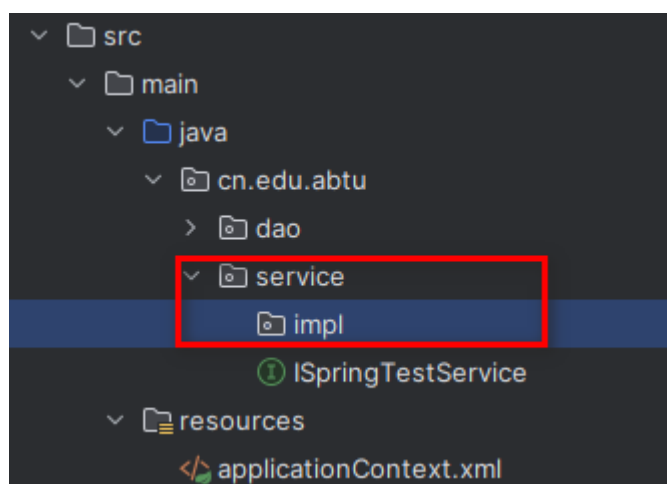
9. 在src->main->java下新建cn.edu.abtu.service包



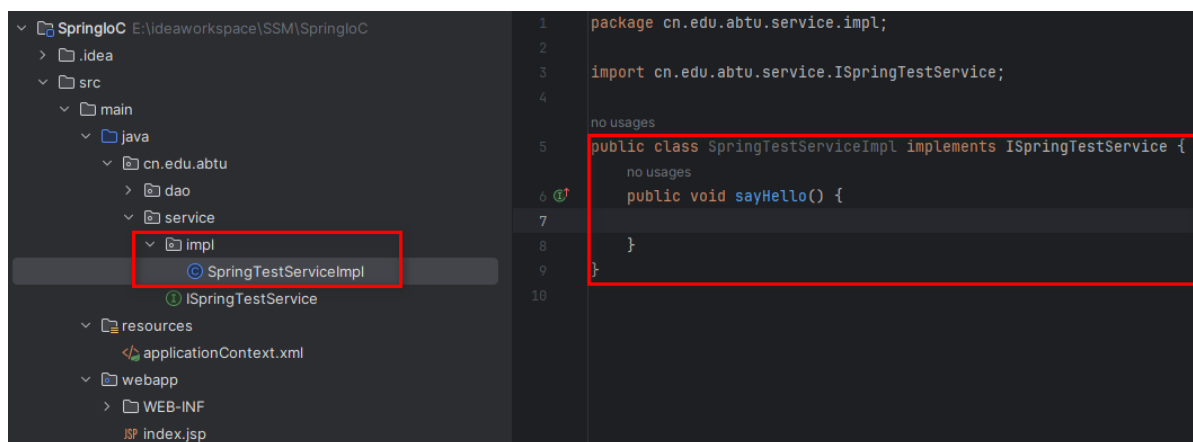
## 10. 在cn.edu.abtu.service包下新建ISpringTestService接口，添加sayHello()方法



## 11. 在src->main->java下新建cn.edu.abtu.service.impl包



## 12. 在cn.edu.abtu.service.impl新建SpringTestServiceImpl类，并实现ISpringTestService接口



13. 通过Service层调用Dao层。即在SpringTestServiceImpl中声明ISpringTestDao的引用，并生成set/get方法

```
no usages
public class SpringTestServiceImpl implements ISpringTestService {

    2 usages
    private ISpringTestDao springTestDao;

    no usages
    public void sayHello() {

    }

    no usages
    public ISpringTestDao getSpringTestDao() {
        return springTestDao;
    }

    no usages
    public void setSpringTestDao(ISpringTestDao springTestDao) {
        this.springTestDao = springTestDao;
    }
}
```

14. 在SpringTestServiceImpl类的sayHello()方法中调用dao层的sayHello()方法

```
public class SpringTestServiceImpl implements ISpringTestService {

    3 usages
    private ISpringTestDao springTestDao;

    no usages
    public void sayHello() {
        springTestDao.sayHello();
    }

    no usages
    public ISpringTestDao getSpringTestDao() {
        return springTestDao;
    }

    no usages
    public void setSpringTestDao(ISpringTestDao springTestDao) {
        this.springTestDao = springTestDao;
    }
}
```

## 15. 在applicationContext.xml中配置要实例化的bean对象

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:task="http://www.springframework.org/schema/task"
       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
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx.xsd
                           http://www.springframework.org/schema/task
                           http://www.springframework.org/schema/task/spring-task.xsd
                           http://www.springframework.org/schema/mvc
                           http://www.springframework.org/schema/mvc/spring-mvc.xsd
                           http://www.springframework.org/schema/util
                           http://www.springframework.org/schema/util/spring-util.xsd">

    <!-- 配置Dao层的实现类bean-->
    <bean id="springTestDao" class="cn.edu.abtu.dao.impl.SpringTestDaoImpl" />

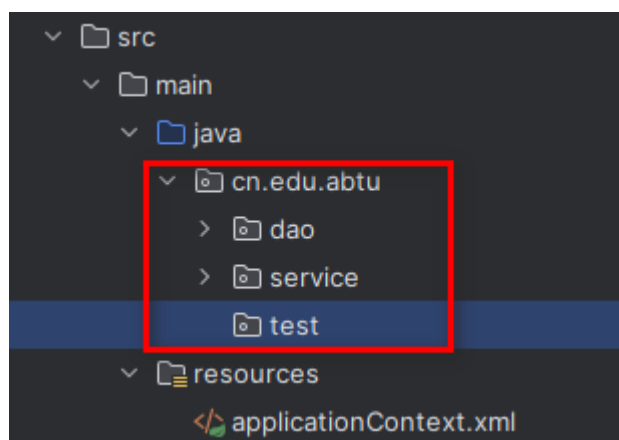
    <!-- 配置service层实现类bean-->
    <bean id="springTestService" class="cn.edu.abtu.service.impl.SpringTestServiceImpl">
        <!-- 注入该bean对象中所依赖的dao对象实例-->
        <property name="springTestDao" ref="springTestDao" />
    </bean>
</beans>
```

每个bean的class属性填写的一定是可以被实例化的具体实现类，不能为接口

ref引用的是所依赖bean的id，一定要与所引用bean的id一样

此处要与SpringTestServiceImpl类中定义的属性（即定义的变量）一定要一样

## 16. 在src->main->java下新建cn.edu.abtu.test包



## 17. 在cn.edu.abtu.test包下新建SpringIocTest类

```
package cn.edu.abtu.test;

public class SpringIocTest {
}
```



## 18. 在SpringIocTest中添加如下测试代码

```
public class SpringIocTest {  
  
    public static void main(String[] args) {  
        ApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
        ISpringTestService springTestService = (ISpringTestService) ac.getBean("springTestService");  
  
        springTestService.sayHello();  
    }  
}
```

此处填写的是resources文件夹下配置文件的名称

此处填写的是applicationContext.xml文件中要获取的service bean对象的id

## 19. 运行代码

```
public class SpringIocTest {  
  
    public static void main(String[] args) {  
        ApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
        ISpringTestService springTestService = (ISpringTestService) ac.getBean("springTestService");  
  
        springTestService.sayHello();  
    }  
}
```

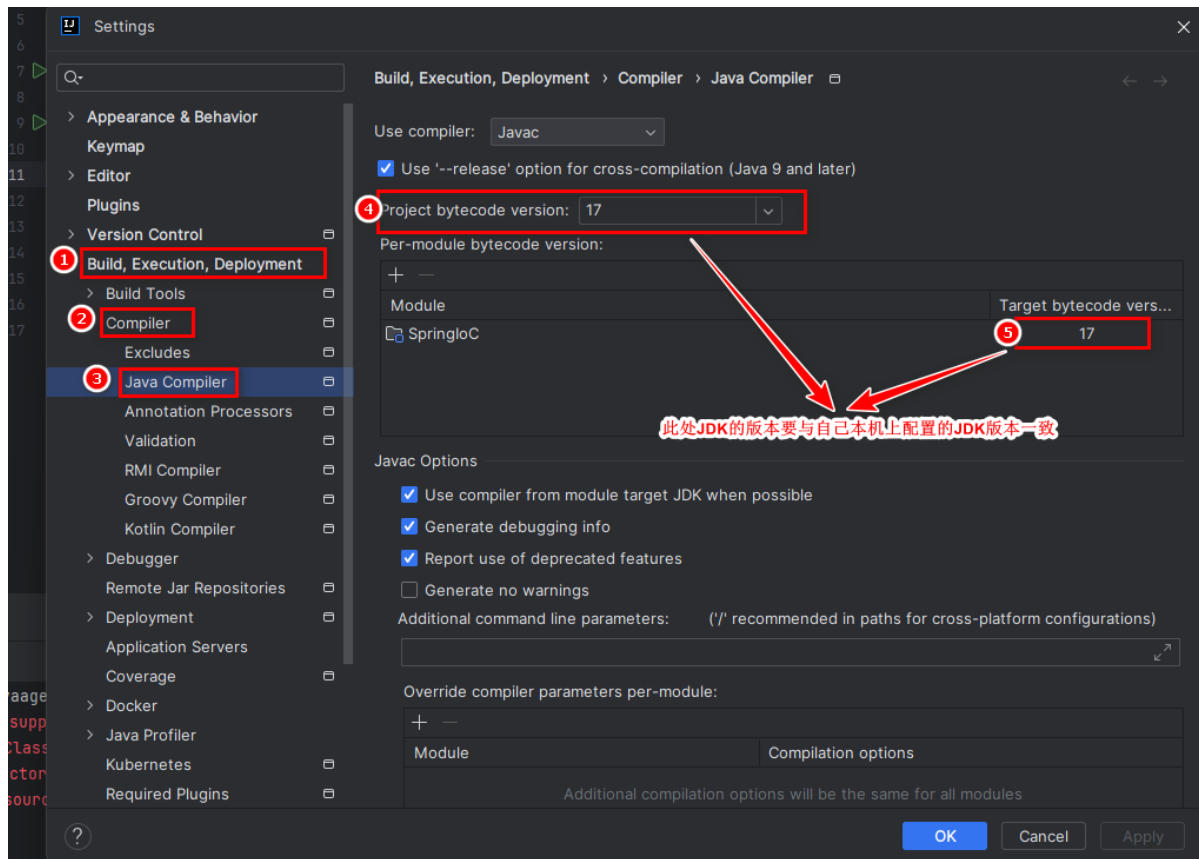
在代码出右键

Run 'SpringIocTest.main()' Ctrl+Shift+F10

## 20. 在控制台出现如下结果，证明项目运行成功

```
Run SpringIocTest x  
"D:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ  
3月 12, 2024 9:56:22 上午 org.springframework.context.support.ClassPathXmlApplicationContext prep  
信息: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@61e717c2: sta  
3月 12, 2024 9:56:22 上午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeaD  
信息: Loading XML bean definitions from class path resource [applicationContext.xml]  
Hello Spring!!!!  
Process finished with exit code 0
```

如果此步出现提示jdk版本不一样的问题，点击File->Settings，在出现的对话框中做如下操作



## (五) 实验总结

通过观察输出结果和自己实验整个过程总结一下自己的学习所得