实验05 Spring框架：Spring的基础应用

**【实验目的及要求】**

1. 掌握Spring框架的基本配置；
2. 掌握使用Spring框架管理bean的基本方法；
3. 熟悉基于Spring框架建立应用程序的基本步骤；
4. 理解控制反转和依赖注入的基本思想；
5. 要求所有回答的文本格式：**五号，宋体、1.5倍行距，保留段单元格背景。**

**【实验步骤】**

**1、引例**

（1）在IDEA中新建一个Maven项目，取名为ex05\_first。

（2）在main/java目录中，新建接口com.tools.Axe，并新建该Axe接口的实现类com.tools.impl.StoneAxe，代码如下：

|  |
| --- |
| package com.tools;  public interface Axe {  void chop(); } |

|  |
| --- |
| package com.tools.impl; import com.tools.Axe;  public class StoneAxe implements Axe {  public void chop() {  System.*out*.println("用石斧砍柴。");  } } |

（3）在main/java目录中，新建类com.service.Person，代码如下：

|  |
| --- |
| package com.service; import com.tools.Axe;  public class Person {  private Axe axe;   public void setAxe(Axe axe) {  this.axe = axe;  }   public void useAxe(){  if (axe == null){  System.*out*.println("斧头不存在！");  return;  }    axe.chop();  } } |

（4）在test/java目录下，新建测试类com.service.PersonTest，代码如下：

|  |
| --- |
| package com.service; import com.tools.Axe; import com.tools.impl.StoneAxe; import org.junit.Test;  public class PersonTest {  @Test  public void useAxeTest(){  Person person = new Person();  Axe axe = new StoneAxe();  person.setAxe(axe);  person.useAxe();  } } |

（5）在main/java目录中，添加一个铁斧类com.tools.impl.SteelAxe，仿照上面的步骤，实现Axe接口。

|  |
| --- |
|  |

（6）修改useAxeTest()测试函数中的代码，使person对象用铁斧砍柴。

|  |
| --- |
|  |

**思考题：**

1. Person和StoneAxe是什么关系？
2. StoneAxe和Axe是什么关系？

|  |
| --- |
| // A： |
| // B： |

**说明**：这个实验有两个作用，其一，回顾包、接口和类的概念及创建方法，其二，阐述一个事实——必须先使用new关键字创建对象后，才能使用该对象。例如，樵夫要砍柴，需要根据斧头设计图纸，先制造一个斧头对象，再使用斧头对象去砍柴。**能否有一个斧头工厂，提供各种型号的斧头（而非斧头图纸）供樵夫选用，从而降低樵夫使用斧头的成本？**

**2、Spring框架的基本配置**

（1）新建Maven项目ex05\_second，在pom.xml文件中添加如下依赖。

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>5.3.16</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.16</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-expression</artifactId>  <version>5.3.16</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>5.3.16</version>  </dependency>  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency> </dependencies> |

（2）在main/java目录中，新建接口com.tools.Arithmetic，接口中包含方法

double calc(double num1, double num2);。再新建com.tools.impl. ArithmeticAdd实现上述接口，完成两个浮点数的加法运算。

|  |
| --- |
| // Arithmetic接口 |
| // ArithmeticAdd类 |

（3）在main/resources目录中新一个config目录，并在该目录中，创建Spring框架的配置文件applicationContext.xml文件，<beans/>元素中的约束如下：

|  |
| --- |
| *<?*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/beans http://www.springframework.org/schema/beans -->   <!-- 你的配置信息 -->* </beans> |

（4）在applicationContext.xml文件中，配置ArithmeticAdd类的bean，bean的名称为arithmetic。配置内容如下：

|  |
| --- |
|  |

（6）在test/java目录中，创建ArithmeticAddTest类，编写test方法，创建Spring容器，并从容器中获得arithmetic对象（bean），再调用arithmetic的calc方法，完成两个浮点数的加法运算和控制台输出。

|  |
| --- |
|  |

（7）给出项目窗口的截图（即IDEA左侧Project窗口），要求展开所有新建的源码和配置文件。

|  |
| --- |
|  |

**3、引例的解决方案**

（1）打开ex05\_first项目，在pom.xml中加入Spring框架的核心依赖，可参考2（2）步骤。

（2）在main/resources目录中，添加beans.xml文件，按2（4）步骤，配置StoneAxe和Person对象（其id或name属性分别为axe和person），并将axe对象注入到person对象中。配置信息如下：

|  |
| --- |
|  |

（3）按2（6）步的方法，修改1（4）步骤的测试代码，获得ApplicationContext容器中的对象person，完成chop()方法的调用。代码如下：

|  |
| --- |
|  |

（4）运行测试方法useAxeTest()，确定结果的有效性。

（5）比较引例中的方法，试阐述Spring框架在bean管理方面的作用。

|  |
| --- |
|  |

**4、引例的进一步扩展（注解方式）**

（1）在第3题的基础上，将XML配置文件方式改为注解方式。要求beans.xml文件中不含<bean/>元素。配置信息如下：

|  |
| --- |
|  |

（2）在类代码中加入注解。

|  |
| --- |
| // Person类 |
| // StoneAxe类 |
| // SteelAxe类 |

（3）运行测试方法useAxeTest()，确定结果的有效性。

**5、Bean实例化方法**

（1）新建Maven项目ex05\_instantiation，在pom.xml中加入Spring框架的核心依赖。

（2）在main/java目录中，新建com.tools.Pen类，代码如下。

|  |
| --- |
| package com.tools;  public class Pen {  void write() {  System.*out*.println("写字");  } } |

（3）在main/resources/config目录中，新建beans.xml文件，给出Pen的bean配置，要求采用构造器实例化bean。配置信息如下。

|  |
| --- |
| *<?*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">  <bean name="pen" class="com.tools.Pen"/> </beans> |

（4）在test/java目录中，新建测试类com.tools.PenTest，检验上述代码与配置的有效性。

|  |
| --- |
| package com.tools;  import org.junit.jupiter.api.Test; import org.springframework.context.ApplicationContext; import org.springframework.context.support.ClassPathXmlApplicationContext;  public class PenTest {  @Test  public void writeTest(){  ApplicationContext applicationContext = new ClassPathXmlApplicationContext("classpath:config/beans.xml");  Pen pen = applicationContext.getBean(Pen.class);  pen.write();  } } |

（5）在在main/java目录中，新建com.factory.StaticBeanFactory类，并在静态方法createPen中实例化Pen。

|  |
| --- |
|  |

（6）修改beans.xml中的配置信息，要求采用静态工厂方式，实例化Pen。再测试静态工厂方法的有效性。配置信息如下：

|  |
| --- |
|  |

（7）在com.factory包中新建DynamicBeanFactory类，并在实例方法createPen中实例化Pen。代码如下：

|  |
| --- |
|  |

（8）修改beans.xml中的配置信息，要求采用实例工厂方式，实例化Pen。再测试实例工厂方法的有效性。配置信息如下：

|  |
| --- |
|  |

**6、基于XML的装配**

（1）新建Maven项目ex05\_assemble，在pom.xml中加入Spring框架的核心依赖。

（2）在main/java目录中，新建com.tools.Pen类，代码如下。

|  |
| --- |
| package com.tools;  public class Pen {  public void write() {  System.*out*.println("写字");  } } |

（3）在main/java目录中，新建接口com.service.Person，代码如下：

|  |
| --- |
| package com.service;  public interface Person {  void usePen(); } |

（4）新建com.service.impl.Student实现类，代码如下。

|  |
| --- |
| package com.service.impl;  import com.service.Person; import com.tools.Pen; import java.util.List; import java.util.Map; import java.util.Set;  public class Student implements Person {  private Pen pen;  private String sno;  private String sname;  private int age;  private String[] interests;  List<String> course;  Set<Double> score;  Map<String,String> otherInfo;*//如address,mobile,qq等* public void usePen() {  System.*out*.print("我开始学习用笔");  pen.write();  }  // 省略了setter和toString方法 } |

（5）在main/resources/config目录中，新建beans.xml文件。在beans.xml文件中增加配置信息，以设值注入方式，实例化Student，要求Student的所有属性都要给出初值。配置信息如下：

|  |
| --- |
| <bean id="pen" class="com.tools.Pen"/> <bean id="student" class="com.service.impl.Student">  <property name="pen" ref="pen"/>  <property name="sname" value="张三"/>  <property name="sno" value="218888999"/>  <property name="age" value="19"/>  <property name="interests">  <array>  <value>蓝球</value>  <value>跑步</value>  <value>爬山</value>  </array>  </property>  <property name="course">  <list>  <value>高等数学</value>  <value>数据结构</value>  <value>计算机网络</value>  </list>  </property>  <property name="score">  <set>  <value>85</value>  <value>92</value>  <value>78</value>  <value>95</value>  </set>  </property>  <property name="otherInfo">  <map>  <entry key="mobile" value="1588889999"/>  <entry key="qq" value="457845"/>  <entry key="address" value="广东省广州市天河区龙洞"/>  </map>  </property> </bean> |

（6）将上一步的设置注入改为构造注入方式，配置信息如下：

|  |
| --- |
|  |

**7、基于注解的装配**

在第6个实验的基础上，删除beans.xml中的所有<bean/>元素，采用注解方法完成bean的装配及注入（List<String>、Set<Double>及Map<String,String>属性等容器内属性不要求注解方式注入。注解方式注入容器类型，通常需要配合属性文件，在Spring Boot中用得多一点）。项目中各部分的代码变化如下：

|  |
| --- |
| // beans.xml中和注解有关的配置信息 |
| // Pen类 |
| // Person类 |
| // Student类 |