第七章习题答案

1.填空题

（1）构造器注入、属性注入

（2）class

（3）scope

（4）singleton

（5）BeanFactory

2.选择题

（1）B

（2）D

（3）A

（4）C

（5）A

3.思考题

（1）

答：Spring支持的作用域如下表所示：

|  |  |
| --- | --- |
| **属性** | **描述** |
| singleton | 单例模式，作用域为singleton的Bean在Spring容器中只会存在一个共享的Bean实例，所有对Bean的请求只要id与Bean的定义相匹配，则只会返回Bean的同一实例 |
| prototype | 每次从容器中调用Bean时，都会产生一个新的Bean实例 |
| request | 一个HTTP请求会产生一个Bean对象，也就是说，每一个HTTP请求都有自己的Bean实例，只在基于Web的Spring ApplicationContext中可用 |
| session | 限定一个Bean的作用域为HTTPsession的生命周期，只有基于Web的Spring ApplicationContext才能使用 |
| global session | 限定一个Bean的作用域为全局HTTPsession的生命周期，只有在Web应用中使用Spring时，该作用域才有效 |

（2）

答：Spring容器可以管理作用域为singleton的Bean的生命周期。对于作用域为prototype的Bean，Spring容器只负责创建实例而不负责跟踪其生命周期。

4.编程题

Ⅰ.创建person对象，定义属性方法

1. package pojo;
2. public class Person {
3. private String name;
4. private int age;
5. public void say() {
6. System.out.println(age);
7. }
8. public Person(String name, int age) {
9. super();
10. this.name = name;
11. this.age = age;
12. }
13. public Person() {
14. super();
15. // TODO Auto-generated constructor stub
16. }
17. public String getName() {
18. return name;
19. }
20. public void setName(String name) {
21. this.name = name;
22. }
23. public int getAge() {
24. return age;
25. }
26. public void setAge(int age) {
27. this.age = age;
28. }
29. @Override
30. public String toString() {
31. return "Person [name=" + name + ", age=" + age + "]";
32. }
34. }

Ⅱ.编写application.xml配置文件信息

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="
5. http://www.springframework.org/schema/beans
6. http://www.springframework.org/schema/beans/spring-beans-3.2.xsd ">
8. <!-- 告诉工厂 给我创建一个person对象 -->
9. <bean id="person" class="pojo.Person">
10. <property name="name" value="zhangsan"></property>
11. <property name="age" value="26"></property>
12. </bean>
13. </beans>

Ⅲ.书写测试类

1. package test;
2. import org.springframework.context.ApplicationContext;
3. import org.springframework.context.support.ClassPathXmlApplicationContext;
4. import pojo.Person;
5. public class TestPerson {
6. public static void main(String[] args) {
7. //使用核心的api初始化工厂 并 从工厂中获取对象
8. ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
9. // 获取对象
10. Person p=(Person) context.getBean("person");
11. System.out.println(p);
12. }
13. }