Examine the below class:

public class Employee

{

public Employee(int empId, int age){

....

}

...

}

Consider Spring Configuration code snippet for the above class:

<bean id="employee" class="Employee">

<constructor-arg \_\_\_\_\_\_\_\_\_\_\_ value="2001"/>

<constructor-arg \_\_\_\_\_\_\_\_\_\_\_ value="24"/>

</bean>

Select correct option(s) to fill the blank lines in the above code for completing the configuration

of the bean to inject values for the properties through constructor injection.

1.

type="int"

type="double"

2.

index="0"

index="1"

3.

index="1"

index="2"

4.

order="0"

order="1"

Examine the below code in Spring configuration file:(Multiple Choice)

<beans>

<bean id="testBean" class="com.cg.beans.Test" >

<property name="testId" value="101"></property>

<property name="testName" value="Java"></property>

<property name="invigilators">

<list>

<ref bean="staff1"/>

<ref bean="staff2"/>

</list>

</property>

</bean>

<bean id="staff1" class="com.cg.beans.Staff">

<property name="staffId" value="4001"></property>

<property name="name" value="Rama"></property>

</bean>

<bean id="staff2" class="com.cg.beans.Staff">

<property name="staffId" value="4002"></property>

<property name="name" value="Sita"></property>

</bean>

</beans>

Choose the appropriate spring beans code which will match with the given configuration code.

1.

package com.cg.beans;

public class Staff {

private int staffId;

private String name;

//getter and setter methods for these properties

}

2.

package com.cg.beans;

public class Staff {

private int staffId;

private String name;

private List<Test> tests;

//getter and setter methods for these properties

}

3.

public class Test {

private int testId;

private String testName;

private List<Staff> invigilators;

//getter and setter methods for these properties

}

4.

public class Test {

private int testId;

private String testName;

private Staff invigilators;

//getter and setter methods for these properties

}

Select the correct statement about spring annotations? (Multiple Choice)

1.

'@Service: This annotation marks classes that implement a part of the business logic of the application.

2.

'@ComponentScan : An equivalent for Spring XML's <context:component-scan/> is provided with

the @ComponentScan annotation.

3.

'@EnableAutoConfiguration annotation will trigger automatic loading of all the beans the application requires.

4.

'@Configuration annotation tells Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context.

(Multiple Choice)

Consider there is a class named as "Student.java" which is annotated with @Component and the class exists in "com.cg.beans" package. Which code inclusion will detect this class?

1.

Include <context:component-scan base-package="com.cg.\*" /> in Spring configuration file

2.

Include <context:component-scan base-package="com.cg.beans" /> in Spring configuration file

3.

Annotated classes will be detected automatically.

4.

Include <context:component-scan base-package="com.cg.Student" /> in web.xml

<bean id="currencyConverter" class="training.Spring.CurrencyConverterImpl">

<property name="exchangeRate" value="44.50" /></bean>

Which type of dependency injection is used in above code?

1.

Setter Injection

2.

Constructor Injection

3.

Interface Injection

4.

No dependency injection is used

Match the following:

1. @Component

2. @Controller

3. @Service

4. @Repository

a. Business logic implementation class will be marked with this annotation

b. Represent a data access logic class

c. Classes annotated with this is referred as spring beans

d. Controller in Spring MVC will be annotated with this.

1.

1 - b, 2 - d, 3 - a, 4 - c

2.

1 - c, 2 - a, 3 - d, 4 - b

3.

1 - c, 2 - d, 3 - b, 4 - a

4.

1 - c, 2 - d, 3 - a, 4 - b

Which of the given option represents loose coupling?

1.

public class Person

{

private Address address;

public Person()

{

this.address = new Address();

}

}

2.

public class Person

{

private Address address;

public Person(Address address)

{

this.address = address;

}

}

3.

Both 0 and 1

4.

none of the above

How would you load a bean from the Spring’s application context?

1.

appContext.getSpringBean(beanName)

2.

appContext.getBean(beanName, beanClass.class)

3.

appContext.getbean(beanName)

4.

appContext.loadBean(beanName)

Which of the following statements is true?

1.

None of these.

2.

ApplicationContext implements BeanFactory.

3.

BeanFactory implements ApplicationContext

4.

ApplicationContext extends BeanFactory.

5.

BeanFactory extends ApplicationContext

Which is the way to provide configuration metadata to spring?

1.

XML Based configuration file.

2.

Annotation based configuration.

3.

Java based configuration.

4.

All of the above.

what is returned if one invokes getBean() specifying the same bean id multiple times when scope of the bean is not defined?

1.

Same class every time

2.

Same instance of the bean

3.

Different instance of the bean

4.

None of them

What will be the output?  
  
public class ShoppingCart   
{  
 private List<Product> items = new ArrayList<Product>();  
 public void addItem(Product item)   
 {  
 items.add(item);  
 }  
 public List<Product> getItems()   
 {  
 return items;  
 }  
}  
   
 <beans ...>  
 <bean id="aaa" class="com.shop.Battery">  
 <property name="name" value="AAA" />  
 <property name="price" value="2.5" />  
 </bean>  
 <bean id="cdrw" class="com.shop.Disc">  
 <property name="name" value="CD-RW" />  
 <property name="price" value="1.5" />  
 </bean>  
 <bean id="dvdrw" class="com.shop.Disc">  
 <property name="name" value="DVD-RW" />  
 <property name="price" value="3.0" />  
 </bean>  
 <bean id="shoppingCart" class="com.shop.ShoppingCart" />  
 </beans>  
   
 import org.springframework.context.ApplicationContext;  
 import org.springframework.context.support.ClassPathXmlApplicationContext;  
 public class Main   
 {  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("beans.xml");  
 Product aaa = (Product) context.getBean("aaa");  
 Product cdrw = (Product) context.getBean("cdrw");  
 Product dvdrw = (Product) context.getBean("dvdrw");  
 ShoppingCart cart1 = (ShoppingCart) context.getBean("shoppingCart");  
 cart1.addItem(aaa);  
 cart1.addItem(cdrw);  
 System.out.println("Shopping cart 1 contains " + cart1.getItems());  
 ShoppingCart cart2 = (ShoppingCart) context.getBean("shoppingCart");  
 cart2.addItem(dvdrw);  
 System.out.println("Shopping cart 2 contains " + cart2.getItems());  
 }  
 }

a) Shopping cart 1 contains (AAA 2.5, CD-RW 1.5)  
Shopping cart 2 contains (AAA 2.5, CD-RW 1.5, DVD-RW 3.0)

b) Shopping cart 1 contains (AAA 2.5, CD-RW 1.5)  
Shopping cart 2 contains (DVD-RW 3.0)

c) BeanCreationException

d) None of the mentioned



In above question if scope of shoppingCart named bean is prototype, then what will be the output?  
What will be the output?

a) Shopping cart 1 contains (AAA 2.5, CD-RW 1.5)  
Shopping cart 2 contains (AAA 2.5, CD-RW 1.5, DVD-RW 3.0)

b) Shopping cart 1 contains (AAA 2.5, CD-RW 1.5)  
Shopping cart 2 contains (DVD-RW 3.0)

c) BeanCreationException

d) None of the mentioned

Which of the given syntax is correct to get a unique instance of a bean each time it is asked for?

1). <bean id=“foo” class=“com.igate.Foo” scope=“prototype” />

2). <bean id=“foo” class=“com.igate.Foo” scope=“singleton” />

3). <bean id=“foo” class=“com.igate.Foo” scope=“request” />

4). <bean id=“foo” class=“com.igate.Foo” scope=“session” />

How to control a bean lifecycle in spring?

a. Using init() method only.  
b. Using InitializingBean and DisposableBean or init() method.

What is the scope of a bean by default in spring?

a. Prototype scope  
b. Singleton scope

Which class represents the IoC container?

a. ApplicationContext  
b. ServletContext

What do you mean by inner bean ?

1.

A <bean/> element inside the <bean/> elements is used to define a so-called inner bean.

2.

A <bean/> element inside the <property/> elements is used to define a so-called inner bean.

3.

A <bean/> element inside the <constructor-arg/> elements is used to define a so-called inner bean.

4.

A <bean/> element inside the <property/> or <constructor-arg/> elements is used to define a so-called inner bean.

public class Wheel{

public Wheel(){

System.out.println("wheel");

{

}

public class Car {

private Wheel wheelObj=null;

public Car(Wheel wheelObj){

this.wheelObj=wheelObj;

}

 public void setWheelObj(Wheel wheel){

  this.wheelObj=wheel;

  }

  public Wheel getWheelObj(){

  return wheelObj;

  }

}

In the above  example, what are correct ways  to write inner bean?

1.

<bean id="wheelBean" class="com.bullraider.Car">  
<constructor-arg>  
<bean name="wheel" class="com.bullraider.Wheel">  
<property name="wheelObj" value="wheel"/>  
</bean>  
</constructor-arg>  
</bean>

2.

<bean id="wheelBean" class="com.bullraider.Car">  
<bean id="carBean" name="car" class="com.bullraider.Wheel">  
<property name="wheelObj" value="wheel"/>  
</bean>  
</bean>

3.

<bean id="wheelBean" class="com.bullraider.Car">  
<property name="wheel">  
<bean name="wheel" class="com.bullraider.Wheel">  
<property name="wheelObj" value="wheel"/>  
</bean>  
</property>  
</bean>

4.

<bean id="wheelBean" class="com.bullraider.Wheel"/>  
<bean id="carBean" name="car" class="com.bullraider.Car">  
<property name="wheelObj" value="wheelBean"/>  
</property>  
</bean>