Spring IOC Set4

What are the metadata Bean Definition objects contain ?

- [ ] class name
- [ ] scope
- [ ] constructor arguments
- [ ] properties
- [ ] autowiring mode
- [ ] dependency checking mode
- [ ] lazy-initialization mode
- [ ] initialization method
- [ ] destruction method
- [x] all the above

What is Bean Factory?

- [x] A BeanFactory is essentially maintains a registry of different beans and their dependencies.
- [x] A BeanFactory enables you to read bean definitions and access them using the bean factory.

- [x] Using just the BeanFactory we can create and read in some bean definitions in the XML format as following

  Resource definition1 = new FileSystemResource("beans-def.xml");

  BeanFactory factory = new XmlBeanFactory(definition1)

- [ ] We can get ApplicationContext and WebApplicationContext object using BeanFactory

Which statements are true about DI(Dependency Injection) ?

DI says your components and services should tightly coupled in code .

- [x] DI says that no need create your objects but describe how they should be created.
- [ ] We should not directly connect your components and services together in code but describe which services are needed by which components in a configuration file
- [x] Inversion of control relies on dependency injection
- [x] We should not directly connect your components and services together in code but describe which services are needed by which components in a configuration file

Which of the following are basic techniques to implement inversion of control?

- [ ] using a factory pattern
- [ ] using a service locator pattern
- [ ] a constructor injection
- [ ] a setter injection
- [ ] an interface injection
- [x] all the above

Which of the following statements are false ?

Constructor-based DI is effected by invoking Setter Injection each representing a dependency

Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments

`@Autowire` annotation annotation on constructor for constructor based autowiring.

- [ ] Constructor-based DI is effected by invoking a constructor with a number of arguments, each representing a dependency
- [ ] We can aslo use @Autowire annotation on constructor for constructor based autowiring.
- [ ] Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments
- [x] Constructor-based DI is effected by invoking Setter Injection each representing a dependency

What are the correct ways to define constructor injection ?

- [ ] <bean id="salute" class="com.bullraider.app.beans.Salutation">
  <constructor-arg>
  <value>Salute</value>
  </constructor-arg>
  </bean>
- [ ] <bean id="salute" class="com.bullraider.app.beans.Salutation">
  <constructor-arg value="Salute">
  </constructor-arg>
  </bean>
- [ ] <bean id="salute" class="com.bullraider.app.beans.Salutation">
  <constructor-arg value="Slute"/>
  </bean>
- [x] All the above

```
public class Wheel{

public Wheel(){

System.out.println("wheel");

{

}



public class Car {

private Wheel wheelObj=null;



  public void setWeelObj(Wheel wheel){

  this.wheelObj=wheel;

  }

  public Wheel getWeelObj(){

 return wheelObj;

  }

}
```

What are the correct ways to write setter injection?

☐
<bean id="wheelBean" class="com.bullraider.Wheel"/>

<bean id="carBean" name="car" class="com.bullraider.Car">

<property name="wheelObj"><ref bean="wheelBean"/></property>

</bean>

☐
<bean id="wheelBean" class="com.bullraider.Wheel"/>

<bean id="carBean" name="car" class="com.bullraider.Car">

<property name="wheelObj" value="wheel"/>

</property>

</bean>

☐
<bean id="wheelBean" class="com.bullraider.Wheel"/>

<bean id="carBean" name="car" class="com.bullraider.Car">

<property name="wheelObj">

<idref bean="wheelBean"/>

</property>

</bean>

☐
<bean id="wheelBean" class="com.bullraider.Wheel"/>

<bean id="carBean" name="car" class="com.bullraider.Car">

<property name="wheelObj" ref bean="wheelBean"/>

</property>

</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?

- [ ]
```
<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>
```

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

- [ ]
```
<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>
```

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

Submit answers