

Quiz Questions

(1) Prototype Bean scope XML

Problem Submissions Doubts

✓ Prototype Bean scope XML
Easy • Score 20/20

Send feedback

Problem statement

Which of the following code snippets correctly defines a bean with prototype scope in Spring?

Options: Pick one correct answer from below

☐

```
<bean id="myBean" class="com.example.Ninja">
  <property name="scope" value="prototype" />
</bean>
```

☒

```
<bean id="myBean" class="com.example.Ninja" scope="prototype">
</bean>
```

☐

```
<bean id="myBean" class="com.example.Ninja">
  <property name="beanScope" value="prototype"/>
</bean>
```

☐

```
<bean id="myBean" class="com.example.Ninja" prototype="true"/>
```

Solution description

- Option A is incorrect, as the scope attribute should not be defined inside a property tag.
- Option B is correct, as the scope attribute can be used directly in the bean tag to define the bean's scope.
- Option C is incorrect as the correct attribute name is scope, not bean scope.
- Option D is incorrect as it defines the bean as a non-singleton bean, which differs from a prototype-scoped bean.

(2) Hello World Bean Scope

Classroom

Beans Scope and Lifecycle
Hello World Bean Scope 3

Problem Submissions Doubts

✓ Hello world bean scope
Easy • Score 20/20

Send feedback

Problem statement

Below are the HelloWorld Class codes and their corresponding XML configuration and Main Class:

```
// Here is the HelloWorld.java
package com.helloworldDemo;

public class HelloWorld {
    private String message;

    public void setMessage(String message){
        this.message = message;
    }

    public void getMessage(){
        System.out.println("Your Message: " + message);
    }
}

// Following is the configuration file application.xml required:
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.springframework.org/XMLSchema"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    // Bean is defined here
    <bean id="helloWorld" class="com.helloworldDemo.HelloWorld" scope="prototype">
        <message>
    </message>
</beans>
```

Options: Pick one correct answer from below

☒ Your Message: I'm Instructor A, Your Message: null

☐ Your Message: null, Your Message: null

☐ Your Message: I'm Instructor A, Your Message: I'm Instructor A

☐ Your Message: null, Your Message: I'm Instructor A

Solution description

- Here in this example, we use the Setter Method for dependency injection. By defining the scope as Prototype in XML configuration, we are ensuring that we need more than one instance of the Bean.
- In the main class, when declaring objA and objB, we use context.getBean() since the scope is prototype every time we call a bean using context.getBean(), we get a new instance of the bean. This is the same as creating two new instances of a Java object.

(3) MyBean Bean Scope III

← Classroom

Beans Scope and Lifecycle
MyBean bean scope III

?

V

Problem Submissions Doubts

Problem statement

Below are the MyBean Class codes and their corresponding XML configuration and Main Class:

```
***
// Here is the MyBean file-
public class MyBean {
    private static int instanceCount = 0;
    private int id;

    public MyBean() {
        id = ++instanceCount;
        System.out.println("Creating an instance of MyBean with id " + id);
    }

    public void doSomething() {
        System.out.println("Doing something with MyBean with id " + id);
    }
}
***
```

```
***
// Following is the configuration file application.xml required:
<?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-3.0.xsd">

    // Bean is defined Here
    <bean id="myBean" class="MyBean" />

</beans>
***
```

```
***
// Following is the MainApp.java file -
public class MainApp {
    public static void main(String[] args) {
        ClassPathApplicationContext context = new
***
```

Options: Pick one correct answer from below

☐ Singleton

☒ Prototype

☐ Request

☐ Session

Solution description

- In the MyBean class has a static variable called instance count, which is incremented every time a new instance of the class is created.
- When the MyBean class's first instance is created, that instance's id field is set to ++instanceCount, equivalent to incrementing instance count by 1 and then assigning the resulting value to id. Therefore, the id of the first instance will be 1.
- When the second instance of the MyBean class is created, the id field of that instance is set to ++instanceCount as well, which is now 2 because the instance count was previously incremented to 1 when the first instance was created.
- This is why the output of the main method shows that the first instance has an ID of 1 and the second instance has an ID of 2.

If the above application gives the following output, what is the scope of MyBean:

```
***
//Output
Creating an instance of MyBean with id 1
Doing something with MyBean with id 1
Creating an instance of MyBean with id 2
Doing something with MyBean with id 2
***
```

(4) Init Method

← Classroom

Beans Scope and Lifecycle
Init Method

?

V

Problem Submissions Doubts

Init method

Easy • Score 20/20

Problem statement

What is the purpose of the "init-method" attribute in a Spring bean definition?

Options: Pick one correct answer from below

☐ To specify a method to be called when the bean is destroyed.

☐ To specify a method to be called after the bean has been instantiated.

☒ To specify a method that should be called after the bean has been initialised.

☐ To specify a method to be called before the bean is initialised.

Solution description

The "init-method" attribute in a Spring bean definition is used to specify a method that should be called after the bean has been initialised.

(5) Destroy Method

← Classroom

Beans Scope and Lifecycle
Bean Destroy Method

?

V

Problem Submissions Doubts

Destroy - method

Easy • Score 20/20

Problem statement

What is the purpose of the "destroy-method" attribute in a Spring bean definition?

Options: Pick one correct answer from below

☐ To specify a method to be called after the bean has been initialised.

☐ To specify a method to be called when the bean is instantiated.

☒ To specify a method to be called when the bean is destroyed.

☐ To specify a method to be called after the bean is destroyed..

Solution description

The "destroy-method" attribute in a Spring bean definition is used to specify a method that should be called when the bean is destroyed.

(6) Bean Lifecycle Code Analysis

← Classroom

Beans Scope and Lifecycle
Bean Lifecycle Code Analysis

Problem Submissions Doubts

✓ Bean Lifecycle_Cod_Analysis
Easy • Score 20/20

Problem statement

Which option is correct for implementing the lifecycle of a bean?

Send feedback

Options: Pick one correct answer from below

☒

```

<bean id="myBean" class="com.example.Ninja" init-method="init"
destroy-method="destroy">
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja">
  <property name="initMethod" value="init" />
  <property name="destroyMethod" value="destroy" />
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja" init-method="init">
  <property name="destroyMethod" value="destroy" />
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja">
  <property name="initMethod" ref="initBean" />
  <property name="destroyMethod" ref="destroyBean" />
</bean>
<bean id="initBean" class="com.example.InitBean"/>
<bean id="destroyBean" class="com.example.DestroyBean" />

```

(7) Bean Lifecycle Implementation

← Classroom

Beans Scope and Lifecycle
Bean Lifecycle Implementation

Problem Submissions Doubts

✓ Bean Lifecycle Implementation
Easy • Score 20/20

Problem statement

What will be the XML Configuration for implementing the bean life cycle of the given code:

Send feedback

```

<<<
public class Ninja implements InitializingBean, DisposableBean {
    private String name;

    public Ninja(String name) {
        this.name = name;
    }

    @Override
    public void initMethod() throws Exception {
        System.out.println("After properties set method called");
    }

    @Override
    public void destroyMethod() throws Exception {
        System.out.println("destroy method called");
    }
}

```

Options: Pick one correct answer from below

☒

```

<bean id="myBean" class="com.example.Ninja" init-method="initMethod"
destroy-method="destroyMethod">
  <constructor-arg value="John" />
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja" init-method="afterPropertiesSet"
destroy-method="destroy">
  <constructor-arg value="John" />
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja" init-method="Ninja"
destroy-method="destroy">
  <constructor-arg value="John" />
</bean>

```

☐

```

<bean id="myBean" class="com.example.Ninja" init-method="initMethod">
  <destroy-method="destroyMethod">
  <constructor-arg value="John" />
</bean>

```

Solution description

The correct way to configure bean lifecycle methods in Spring XML configuration is to use the init-method and destroy-method attributes in the element. Option A correctly sets the init-method and destroy-method attributes to "initMethod" and "destroyMethod."