Q. If we create two bean with different-2 scope and then we inject one bean into other bean then what will be the scope of injecting bean. We are discussing here two case.

**First case:-**

**Q.1**-One bean has prototype scope(which bean is going to be injected i.e **Engine**) and another bean has singleton scope(In which bean will be injected i.e. **Car**).

Then what will be the scope of **Engine** Bean.

**Ans:-** Car has singleton Scope when Spring container creates object for car it also inject Engine bean.

Car has singleton scope so it is created one time that’s why Engine’s id will be same.

**Car.java**

package com.doj.spring;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.config.ConfigurableBeanFactory;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="car" class="com.doj.spring.Car"/>

\*/

@Component

//@Component("car") //We use this to remove ambiguity of beans we have two same car in same base but it will create ambiguity that's why we give name to every bean.

package(com.doj.spring)

@Scope(ConfigurableBeanFactory.SCOPE\_SINGLETON)

public class Car {

@Autowired

Engine engine;

public Car() {

super();

}

public Engine getEngine(){

return engine;

}

}

**Engine.java**

package com.doj.spring;

import org.springframework.beans.factory.config.ConfigurableBeanFactory;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="engine" class="com.doj.spring.Engine"/>

\*/

@Component

//@Scope(ConfigurableBeanFactory.SCOPE\_SINGLETON)

@Scope(ConfigurableBeanFactory.SCOPE\_PROTOTYPE)

public class Engine {

final String engineName="FORD";

public String getEngineName() {

return engineName;

}

}

JavaBasedConfiguration is here

**SpringConfiguration.java**

package com.doj.spring;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

@Configuration

/\*

\* @Configuration annotation is used to define java based configuration this works like spring.xml

\*/

@ComponentScan(basePackages={"com.doj.spring"})

//@ComponentScan by default it search component in current package but we can also define another package also like below

//@ComponentScan(basePackages={"com.org"})//can also define more then one package here because this String array type

public class SpringConfiguration {

}

**CarDemo.java**

**package** com.doj.spring;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**public** **class** CarDemo {

**public** **static** **void** main(String[] args) {

//ClassPathXmlApplicationContext is using for XML based configuration

//AbstractApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");

//AnnotationConfigApplicationContext is using for JAVA based configuration

AbstractApplicationContext context = **new** AnnotationConfigApplicationContext(SpringConfiguration.**class**);

Car car = (Car) context.getBean("car");

System.*out*.println(car+" | "+car.getEngine());

Car car1 = (Car) context.getBean("car");

System.*out*.println(car1+" | "+car1.getEngine());

context.close();

}

}

**Output:-**

com.doj.spring.Car@afc191 | com.doj.spring.Engine@12fc6eb

com.doj.spring.Car@afc191 | [com.doj.spring.Engine@12fc6eb](mailto:com.doj.spring.Engine@12fc6eb)

**Second case:-**

**Q.2**-One bean has singleton scope(which bean is going to be injected i.e **Engine**) and another bean has prototype scope(In which bean will be injected i.e. **Car**).

Then what will be the scope of **Engine** Bean.

Ans:-

Singleton scope means one time it is created pr Spring Container then same is used every time. So first time it is injected to car bean . and second time also same(Engine) bean will be injected because Engine has singleton scope.

**Car.java**

package com.doj.spring;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.config.ConfigurableBeanFactory;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="car" class="com.doj.spring.Car"/>

\*/

@Component

//@Component("car") //We use this to remove ambiguity of beans we have two same car in same base package(com.doj.spring) but it will create ambiguity that's why we give name to every bean

@Scope(ConfigurableBeanFactory.SCOPE\_PROTOTYPE)

public class Car {

@Autowired

Engine engine;

public Car() {

super();

}

public Engine getEngine(){

return engine;

}

}

**Engine.java**

package com.doj.spring;

import org.springframework.beans.factory.config.ConfigurableBeanFactory;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id="engine" class="com.doj.spring.Engine"/>

\*/

@Component

@Scope(ConfigurableBeanFactory.SCOPE\_SINGLETON)

public class Engine {

final String engineName="FORD";

public String getEngineName() {

return engineName;

}

}

JavaBasedConfiguration file

package com.doj.spring;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

@Configuration

/\*

\* @Configuration annotation is used to define java based configuration this works like spring.xml

\*/

@ComponentScan(basePackages={"com.doj.spring"})

//@ComponentScan by default it search component in current package but we can also define another package also like below

//@ComponentScan(basePackages={"com.org"})//can also define more then one package here because this String array type

public class SpringConfiguration {

}

**CarDemo.java**

**package** com.doj.spring;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**public** **class** CarDemo {

**public** **static** **void** main(String[] args) {

//ClassPathXmlApplicationContext is using for XML based configuration

//AbstractApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");

//AnnotationConfigApplicationContext is using for JAVA based configuration

AbstractApplicationContext context = **new** AnnotationConfigApplicationContext(SpringConfiguration.**class**);

Car car = (Car) context.getBean("car");

System.*out*.println(car+" | "+car.getEngine());

Car car1 = (Car) context.getBean("car");

System.*out*.println(car1+" | "+car1.getEngine());

context.close();

}

}

Output:-

com.doj.spring.Car@12fc6eb | com.doj.spring.Engine@60b431

com.doj.spring.Car@12679d6 | com.doj.spring.Engine@60b431