1.

<?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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="..." class="...">

<!-- collaborators and configuration for this bean go here -->

</bean>

<bean id="..." class="...">

<!-- collaborators and configuration for this bean go here -->

</bean>

<!-- more bean definitions go here -->

</beans>

2.

// create and configure beans

ApplicationContext context = new ClassPathXmlApplicationContext("services.xml", "daos.xml");

// retrieve configured instance

PetStoreService service = context.getBean("petStore", PetStoreService.class);

// use configured instance

List<String> userList = service.getUsernameList();

3

@Configuration

@ComponentScan(basePackages = "lk.ijse.spring.pojo")

public class AppConfig {

}

4

@ComponentScan(basePackages = "lk.ijse.spring.pojo")

5

@ComponentScan(basePackages = {"lk.ijse.spring.pojo","lk.ijse.spring.controller"})

6

@ComponentScan(basePackageClasses = CustomerController.class)

7

@Component("TestPojo")

public class MyPojo {

}

9

BasicDataSource bean1 = ctx.getBean(BasicDataSource.class);

BasicDataSource bean2 = (BasicDataSource) ctx.getBean("test");

8

@Configuration

@ComponentScan(basePackages = "lk.ijse.spring.pojo")

public class AppConfig {

   public AppConfig(){

       System.*out*.println("Hi I'm AppConfig");

   }

   @Bean

   public BasicDataSource test(){

       return new BasicDataSource();

   }

}

10

@Bean("sample")

public BasicDataSource test (){

   return new BasicDataSource();

}

11

BasicDataSource test = (BasicDataSource) ctx.getBean("sample");

12

MyPojo myPojo1 = ctx.getBean(MyPojo.class);

13

MyPojo myPojo2 = (MyPojo) ctx.getBean("myPojo");

14

MyPojo myPojo1 = ctx.getBean(MyPojo.class);

MyPojo myPojo2 = (MyPojo) ctx.getBean("myPojo");

System.*out*.println(myPojo1); // lk.ijse.spring.pojo.MyPojo@8b96fde

System.*out*.println(myPojo2); // [lk.ijse.spring.pojo.MyPojo@8b96fde](mailto:lk.ijse.spring.pojo.MyPojo@8b96fde)

15

@Scope(ConfigurableBeanFactory.*SCOPE\_SINGLETON*)

Or

@Scope("singleton")

16

@Scope(ConfigurableBeanFactory.*SCOPE\_PROTOTYPE*)

Or

@Scope("prototype")

17

Runtime.*getRuntime*().addShutdownHook(new Thread(new Runnable() {

   @Override

   public void run() {

       ctx.close();

   }

}));

18

public class AppInitializer {

   public static void main(String[] args) {

       AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();

       ctx.register(AppConfig.class);

       ctx.refresh();

       ctx.registerShutdownHook();

   }

}

19

@Component

public class PojoOne implements BeanNameAware, BeanFactoryAware, ApplicationContextAware, InitializingBean , DisposableBean {

   public PojoOne(){

       System.*out*.println("Pojo1: Instantiate");

   }

   @Override

   public void setBeanName(String name) {

       System.*out*.println("Pojo1: Bean Name Aware :"+name);

   }

   @Override

   public void setBeanFactory(BeanFactory beanFactory) throws BeansException {

       System.*out*.println("Pojo1: Bean Factory Aware");

   }

   @Override

   public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {

       System.*out*.println("Pojo1: Application Context Aware");

   }

   @Override

   public void afterPropertiesSet() throws Exception {

       System.*out*.println("Pojo1: Initializing Bean");

   }

   @Override

   public void destroy() throws Exception {

       System.*out*.println("Pojo1: Disposable Bean");

   }

}

20

A screen shot of a computer

Description automatically generated

21

public class Engine {

private String type;

public Engine(String type) {

this.type = type;

}

public String getType() {

return type;

}

}

public class Car {

private Engine engine;

// Constructor Injection

public Car(Engine engine) {

this.engine = engine;

}

public void drive() {

System.out.println("Driving car with " + engine.getType() + " engine.");

}

}

22

XML Configuration:

<bean id="engine" class="com.example.Engine">

<constructor-arg value="V8"/>

</bean>

<bean id="car" class="com.example.Car">

<constructor-arg ref="engine"/>

</bean>

23

Java Code to Retrieve Beans:

ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

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

car.drive();

24

public class Car {

private Engine engine;

// Setter Injection

public void setEngine(Engine engine) {

this.engine = engine;

}

public void drive() {

System.out.println("Driving car with " + engine.getType() + " engine.");

}

}

25

<bean id="engine" class="com.example.Engine">

<property name="type" value="V8"/>

</bean>

<bean id="car" class="com.example.Car">

<property name="engine" ref="engine"/>

</bean>

26

ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

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

car.drive();

27

public class Car {

@Autowired

private Engine engine;

public void drive() {

System.out.println("Driving car with " + engine.getType() + " engine.");

}

}

28

public class Boy{

   // Property injection

   GoodGirlAgreement girl = new Girl();

   public void chatWithGirl() {

       girl.chat();

   }

}

29

public interface GoodGirlAgreement {

   public void chat();

}

30

public class Girl implements GoodGirlAgreement {

   @Override

   public void chat(){

       System.*out*.println("Girl Says : Hi");

   }

}

31

public class AppInitializer {

   public static void main(String[] args) {

       Boy boy = new Boy();

       boy.chatWithGirl();

   }

}

32

@Component

public class Boy{

   // Property injection

   @Autowired

   GoodGirlAgreement girl;

   public void chatWithGirl() {

       girl.chat();

   }

}

33

@Component

public class Boy{

   GoodGirlAgreement girl;

   // Constructor injection

   @Autowired

   public Boy(GoodGirlAgreement girl) {

       this.girl = girl;

   }

   public void chatWithGirl() {

       girl.chat();

   }

}

34

@Component

public class Boy {

   GoodGirlAgreement girl;

   // Setter method injection

   @Autowired

   public void setGirl(GoodGirlAgreement girl) {

       this.girl = girl;

   }

   public void chatWithGirl() {

       girl.chat();

   }

}

35

public interface Inject {

   void setInject(GoodGirlAgreement girl);

}

@Component

public class Boy implements Inject{

   GoodGirlAgreement girl;

   // Interface trough injection

   @Autowired

   @Override

   public void setInject(GoodGirlAgreement girl) {

       this.girl = girl;

   }

   public void chatWithGirl() {

       girl.chat();

   }

}

36

@Component

@Primary

public class GirlOne implements GoodGirlAgreement {

   @Override

   public void chat() {

       System.*out*.println("GirlOne Says : Hi");

   }

}

37

@Component

public class Boy{

   @Autowired

   @Qualifier("girlOne")

   GoodGirlAgreement girl;

   public void chatWithGirl() {

       girl.chat();

   }

}

38

<bean id="parentBean" class="com.example.MyBean" abstract="true">

<property name="property1" value="Common Value"/>

<property name="property2" value="Default Value"/>

</bean>

39

<bean id="childBean1" parent="parentBean">

<property name="property2" value="Overridden Value"/>

<property name="property3" value="Unique Value 1"/>

</bean>

<bean id="childBean2" parent="parentBean">

<property name="property2" value="Different Overridden Value"/>

<property name="property3" value="Unique Value 2"/>

</bean>

40

@Configuration

@ComponentScan(basePackages = "lk.ijse.spring.pojo")

public class AppConfig {

   @Bean

   public PojoTwo pojoTwo() {

       // inter-bean dependencies invocation

       PojoThree pojoThree1 = pojoThree();

       System.*out*.println(pojoThree1);

       PojoThree pojoThree2 = pojoThree();

       System.*out*.println(pojoThree2);

       return new PojoTwo();

   }

   @Bean

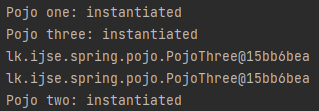
   public PojoThree pojoThree() {

       return new PojoThree();

   }

}

41



42

@Component

public class PojoOne {

   @Bean

   public PojoTwo pojoTwo() {

       PojoThree pojoThree1 = pojoThree();

       System.*out*.println(pojoThree1);

       PojoThree pojoThree2 = pojoThree();

       System.*out*.println(pojoThree2);

       return new PojoTwo();

   }

   @Bean

   public PojoThree pojoThree() {

       return new PojoThree();

   }

}

43



44