/\*  
1. about the beans  
 @Component --- put this line above the class which we want to declare as a beans.  
  
2. about dependencies  
 @Autowired --- put this is above the variable declaration to make and assigned depended object automatically.  
  
3. Where to scan for beans

---if not declare than it is only scan base package using @SpringBootApplication   
 -- this is used above main class to show san all classes in same package  
 @ComponentScan(basePackages={"",""}) --this is used to show if packaged is different

@primary – is used to show which Component is used first in the case there are multiple beans of same type of class.

Ex. Dog , Cat  
  
 \*/

@SpringBootApplication  
//@ComponentScan(basePackages = {"com.lcwd.core","test","xyz"})  
public class SpringCoreConceptsApplication {  
 public static void main(String[] args) {  
 //Animal animal = new Dog();  
 //Animal animal = new Cat();  
 //Person p = new Person(animal);  
  
 ApplicationContext context =SpringApplication.*run*(SpringCoreConceptsApplication.class, args);  
 Person personBean = context.getBean(Person.class);  
 personBean.playWithAnimal();  
  
 }  
}

@Component  
public class Dog implements Animal {  
 @Override  
 public void play() {  
 System.*out*.println("Dag is Playing");  
 }  
}

@Component  
public class Person {  
  
 @Autowired  
 Animal animal;  
  
 public Person(Animal animal) {  
 this.animal = animal;  
 }  
  
 public void playWithAnimal(){  
 // Using Animal  
 animal.play();  
 }  
}

@Component  
public class Person {  
  
 @Autowired  
 @Qualifier("dog")  
 Animal animal;  
  
// public Person(Animal animal) {  
// this.animal = animal;  
// }  
  
 public void playWithAnimal(){  
 // Using Animal  
 animal.play();  
 }  
}

@Autowired  
 @Qualifier("dog")  
 Animal animal;  
With this style we can’t use constructor method.

For this is issue use below style. --- Use @Autowired above constructor method and use @Qualifier at parameter

@Component  
public class Person {  
  
 //@Autowired  
 //@Qualifier("cat")  
 Animal animal;  
  
 @Autowired  
 public Person(@Qualifier("cat") Animal animal) {  
 this.animal = animal;  
 }  
 public void playWithAnimal(){  
 // Using Animal  
 animal.play();  
 }  
}

@Component("dog")  
public class Dog implements Animal {  
 @Override  
 public void play() {  
 System.*out*.println("Dag is Playing");  
 }  
}

@Component("cat")  
//@Primary  
public class Cat implements Animal{  
 @Override  
 public void play() {  
 System.*out*.println("Cat is Playing");  
 }  
}

We can write like below also

Component  
@Qualifier("cat")

@Component  
@Qualifier("cat")  
//@Primary  
public class Cat implements Animal{  
 @Override  
 public void play() {  
 System.*out*.println("Cat is Playing");  
 }  
}

If once we used below style above class than we can get that object by name with the help of context.getbean()

@Component(“cat”)

or

Component  
@Qualifier("cat")

@SpringBootApplication  
//@ComponentScan(basePackages = {"com.lcwd.core","test","xyz"})  
public class SpringCoreConceptsApplication {  
 public static void main(String[] args) {  
 //Animal animal = new Dog();  
 //Animal animal = new Cat();  
 //Person p = new Person(animal);  
  
 ApplicationContext context =SpringApplication.*run*(SpringCoreConceptsApplication.class, args);  
 Person personBean = context.getBean(Person.class);  
 personBean.playWithAnimal();  
  
 Animal cat = context.getBean("cat", Animal.class);  
 Animal dog = context.getBean("dog", Animal.class);  
 cat.play();  
 dog.play();  
  
  
 }  
}

**Without SprintBoot How we can make spring project.**

1. Create Simple maven project
2. Set archetype = org.apache.maven.archetypes:maven-archetype-quickstart
3. Add spring formwork dependency in pom.xml

* Add spring-core and spring-context Dependency. So Maven will automatically load all required jar file
* <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->  
  <dependency>  
   <groupId>org.springframework</groupId>  
   <artifactId>spring-core</artifactId>  
   <version>6.0.8</version>  
  </dependency>  
  <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
  <dependency>  
   <groupId>org.springframework</groupId>  
   <artifactId>spring-context</artifactId>  
   <version>6.0.8</version>  
  </dependency>

1. Use Annotation and create appConfig related class to make spring.

public class App   
{  
 public static void main( String[] args )  
 {  
 System.*out*.println( "Application Started" );  
  
 //We have to create object of Application context  
  
 /\* For XML based configuration \*/  
 //ApplicationContext context =new ClassPathXmlApplicationContext();  
  
 /\* For Annotation based Configuration \*/  
 ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);  
 //CartService cartService1= (CartService) context.getBean("cartService1"); // We can write this way also  
 CartService cartService1= context.getBean("cartService1", CartService.class);  
 cartService1.createCart();  
  
 UserService bean = context.getBean(UserService.class);  
 bean.saveUser();  
  
 UserService bean1 = context.getBean(UserService.class);  
 bean1.saveUser();  
  
 OrderService bean2 = context.getBean(OrderService.class);  
 bean2.createOrder();  
  
 }  
}

@Configuration  
@ComponentScan("com.lcwd.withoutboot")  
public class AppConfig {  
 @Bean("cartService1")  
 public CartService CartService(){  
 return new CartService();  
 }  
// @Bean  
// public VeiewResolver viewResolver(){  
// VeiewResolver v= new ViewResolver();  
// return v;  
// }

@Component  
public class UserService {  
 public void saveUser(){  
 System.*out*.println("User is add Sucessfully");  
 }  
}

@Component  
public class OrderService {  
 public void createOrder(){  
 System.*out*.println("one Order is created");  
 }  
}

public class CartService {  
 public void createCart(){  
 System.*out*.println("one cart id Added");  
 }  
}

