**Spring IOC**

1. Truy cập “ref” giữa các bean

**Cùng file.XML**

**Spring.xml**

<bean id=*"dogBean"* class=*"tctav.com.demo.farm.Dog"* />

<bean id=*"catBean"* class=*"tctav.com.demo.farm.Cat"* />

<bean id=*"houseBean"* class=*"tctav.com.demo.farm.House"*>

<property name=*"animal1"*>

<ref local=*"dogBean"* />

</property>

<property name=*"animal2"*>

<ref local=*"catBean"* />

</property>

</bean>

**Khác file.XML**

**Spring1.xml**

<bean id=*"dogBean"* class=*"tctav.com.demo.farm.Dog"* />

<bean id=*"catBean"* class=*"tctav.com.demo.farm.Cat"* />

**Spring2.xml**

<bean id=*"houseBean"* class=*"tctav.com.demo.farm.House"*>

<property name=*"animal1"*>

<ref bean=*"dogBean"* />

</property>

<property name=*"animal2"*>

<ref bean=*"catBean"* />

</property>

</bean>

<bean id=*"dogBean"* class=*"tctav.com.demo.farm.Dog"* />

<bean id=*"catBean"* class=*"tctav.com.demo.farm.Cat"* />

<bean id=*"houseBean"* class=*"tctav.com.demo.farm.House"*>

<property name=*"animal1"* ref=*"dogBean"* />

<property name=*"animal2"* ref=*"catBean"* />

</bean>

**Dùng chung không phân biệt chung hay riêng file.XML**

1. Các cách inject 1 value trong bean

**Cách thông thường**:

<bean id=*"FileNameGenerator1"* class=*"tctav.com.demo.model.FileNameGenerator"*>

<property name=*"name"*>

<value>mhnguyen1</value>

</property>

<property name=*"type"*>

<value>txt</value>

</property>

</bean>

**Cách rút** **gọn**:

<bean id=*"FileNameGenerator2"* class=*"tctav.com.demo.model.FileNameGenerator"*>

<property name=*"name"* value=*"mhnguyen2"* />

<property name=*"type"* value=*"exe"* />

</bean>

**Cách rút gọn sử dụng “p”**

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"*>

<bean id=*"FileNameGenerator3"* class=*"tctav.com.demo.model.FileNameGenerator"* p:name=*"mhnguyen3"* p:type=*"pdf"* />

</bean>

**txtav.com.demo.model.FileNameGenerator**

**public** **class** FileNameGenerator {

**private** String name;

**private** String type;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

}

1. Load mutiple Spring bean trong file cấu hình

**root-context.xml**

<?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-2.0.xsd"*>

<import resource=*"module/farm.xml"*/>

</beans>

**module/farm.xml**

<?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-2.0.xsd"*>

<bean id=*"dogBean"* class=*"tctav.com.demo.farm.Dog"* />

<bean id=*"catBean"* class=*"tctav.com.demo.farm.Cat"* />

<bean id=*"houseBean"* class=*"tctav.com.demo.farm.House"*>

<property name=*"animal1"* ref=*"dogBean"* />

<property name=*"animal2"* ref=*"catBean"* />

</bean>

</bean>

1. Map bean và class qua “property” và “constructor-arg”

**Person.java**

**public** **class** Person {

**private** String name;

**private** String address;

**private** **int** age;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**Customer.java**

**public** **class** Customer {

**private** Person person;

**public** Customer(Person person) {

**this**.person = person;

}

**public** Person getPerson() {

**return** person;

}

**public** **void** setPerson(Person person) {

**this**.person = person;

}

}

**Bean.xml**

<!—- Map thông qua property -->

<bean id=*"customerBean"* class=*"tctav.com.demo.innerBean.Customer"*>

<property name=*"person"* ref=*"personBean"* />

</bean>

<bean id=*"personBean"* class=*"tctav.com.demo.innerBean.Person"*>

<property name=*"name"* value=*"mhnguyen"* />

<property name=*"address"* value=*"326 Vo Van Kiet"* />

<property name=*"age"* value=*"27"*/>

</bean>

<!—- Map thông qua property -->

<bean id=*"customerBean"* class=*"tctav.com.demo.innerBean.Customer"*>

<property name=*"person"*>

<bean class=*"tctav.com.demo.innerBean.Person"*>

<property name=*"name"* value=*"mhnguyen"* />

<property name=*"address"* value=*"326 Vo Van Kiet"* />

<property name=*"age"* value=*"27"*/>

</bean>

</property>

</bean>

<!—- Map thông qua contructor -->

<bean id=*"customerBean"* class=*"tctav.com.demo.innerBean.Customer"*>

<constructor-arg>

<bean class=*"tctav.com.demo.innerBean.Person"*>

<property name=*"name"* value=*"mhnguyen"* />

<property name=*"address"* value=*"326 Vo Van Kiet"* />

<property name=*"age"* value=*"27"*/>

</bean>

</constructor-arg>

</bean>

Một vài ví dụ tham khảo thêm:

*Interface*: Animal

**public** **interface** Animal {

**public** String makeSound();

}

*Class*: Dog, Cat implement interface Animal

**public** **class** Dog **implements** Animal {

@Override

**public** String makeSound() {

**return** "Dog makes sound";

}

}

* **Map Bean thông qua Set & Get**

<bean id=*"dogBean"* class=*"tctav.com.demo.farm.Dog"* />

<bean id=*"catBean"* class=*"tctav.com.demo.farm.Cat"* />

<bean id=*"houseBean"* class=*"tctav.com.demo.farm.House"*>

<property name=*"animal1"* ref=*"dogBean"* />

<property name=*"animal2"* ref=*"catBean"* />

</bean>

* **Map Bean thông qua Constructor có đầu vào là một interface**

<bean id=*"dogBean"* class=*"com.demo.Dog"*/>

<bean id=*"catBean"* class=*"com.demo.Cat"* />

<bean id=*"houseBean"* class=*"com.demo.House"*>

<constructor-arg ref=*"dogBean"*/>

<constructor-arg ref=*"catBean"*/>

</bean>

**public** **class** House {

Animal animal1;

Animal animal2;

**public** House(Animal animal2, Animal animal1){

**this**.animal2 = animal2;

**this**.animal1 = animal1;

}

}

* **Map Bean thông qua Constructor có đầu vào là một Class**

<bean id=*"houseBean"* class=*"com.demo.House"*>

<constructor-arg>

<bean class=*"com.demo.Dog"*></bean>

</constructor-arg>

</bean>

**public** **class** House {

Animal animal1;

**public** House(Dog dog){

**this**.animal1 = dog;

}

}

1. Singleton scope và Prototype scope

**Example sử dụng Bean.xml**

**CustomerService.java**

**public** **class** CustomerService {

**private** String message;

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

@Override

**public** String toString() {

**return** **this**.message;

}

}

**Bean.xml (scope singleton và scope prototype)**

<bean id=*"customerService"* class=*"ctcav.com.demo.scope.CustomerService"*/>

<bean id=*"customerService"* class=*"ctcav.com.demo.scope.CustomerService"* scope=*"prototype"* />

**Main.java**

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("Bean.xml");

CustomerService customer1 = (CustomerService) context.getBean("customerService");

customer1.setMessage("customer 1");

System.***out***.println(customer1.toString());

CustomerService customer2 = (CustomerService) context.getBean("customerService");

System.***out***.println(customer2.toString());

}

**Output**

**Scope singleton**:

customer 1

customer 1

**Scope prototype**:

customer 1

null

**Example sử dụng Anotation**

**CustomerService.java**

@Service

@Scope("prototype") // Config prototype scope

**public** **class** CustomerService {

**private** String message;

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

@Override

**public** String toString() {

**return** **this**.message;

}

}

**Bean.xml (bật tính năng tự động scan các thành phần)**

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"*>

<context:component-scan base-package=*"ctcav.com.demo"* />

</beans>

1. Collections in Spring (List, Set, Map, Properties)

**CustomerCollections.java**

**public** **class** CustomerCollections {

**private** List<Object> lists;

**private** Set<Object> sets;

**private** Map<Object,Object> maps;

**private** Properties pros;

// set, get properties

@Override

**public** String toString() {

String getLists = "getLists: " + **this**.getLists().toString();

String getSets = "getSets: " + **this**.getSets().toString();

String getMaps = "getMaps: " + **this**.getMaps().toString();

String getPros = "getPros: " + **this**.getPros().toString();

**return** getLists + "\n" + getSets + "\n" + getMaps + "\n" + getPros;

}

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("bean.xml");

CustomerCollections cols = (CustomerCollections) context.getBean("customercollections");

System.***out***.println(cols.toString());

}

}

**Bean.xml**

<bean id=*"customercollections"* class=*"ctcav.com.demo.collections.CustomerCollections"*>

<!-- java.util.List -->

<property name=*"lists"*>

<list>

<value>1</value>

<ref bean=*"personBean"*/>

<bean class=*"ctcav.com.demo.collections.Person"*>

<property name=*"name"* value=*"mhnguyen2"* />

<property name=*"address"* value=*"123 Vo Van Kiet"* />

<property name=*"age"* value=*"28"* />

</bean>

</list>

</property>

<!-- java.util.Set -->

<property name=*"sets"*>

<set>

<value>1</value>

<ref bean=*"personBean"*/>

<bean class=*"ctcav.com.demo.collections.Person"*>

<property name=*"name"* value=*"mhnguyen2"* />

<property name=*"address"* value=*"123 Vo Van Kiet"* />

<property name=*"age"* value=*"28"* />

</bean>

</set>

</property>

<!-- java.util.Map -->

<property name=*"maps"*>

<map>

<entry key=*"key 1"* value=*"value 1"*/>

<entry key=*"key 2"* value=*"value 2"*/>

<entry key=*"key 3"*>

<ref bean=*"personBean"*/>

</entry>

<entry key=*"key 4"*>

<bean class=*"ctcav.com.demo.collections.Person"*>

<property name=*"name"* value=*"mhnguyen2"* />

<property name=*"address"* value=*"123 Vo Van Kiet"* />

<property name=*"age"* value=*"28"* />

</bean>

</entry>

</map>

</property>

<!-- java.util.properties -->

<property name=*"pros"*>

<props>

<prop key=*"admin"*>123</prop>

<prop key=*"support"*>456</prop>

</props>

</property>

</bean>

**Example for Web**

**Controller.java**

@Autowired **private** ArrayList<String> arrayListBean;

@Autowired **private** HashMap<Integer, String> hasmapBean;

**Bean.xml**

<!-- ArrayList -->

<bean id=*"arrayListBean"* class=*"java.util.ArrayList"*>

<constructor-arg>

<list>

<value>1</value>

<value>2</value>

<value>3</value>

</list>

</constructor-arg>

</bean>

<!-- HashMap -->

<bean id=*"hasmapBean"* class=*"java.util.HashMap"*>

<constructor-arg>

<map key-type=*"java.lang.Integer"* value-type=*"java.lang.String"*>

<entry key=*"1"* value=*"value 1"*/>

<entry key=*"2"* value=*"value 2"*/>

<entry key=*"3"* value=*"value 3"*/>

</map>

</constructor-arg>

</bean>

ListFactoryBean, SetFactoryBean, MapFactoryBean

**CustomerFactory.java**

**public** **class** CustomerFactory {

**private** List lists;

**private** Set sets;

**private** Map maps;

// Set, get properties

@Override

**public** String toString() {

String getLists = lists != **null** ? "lists: " + lists.toString() : "";

String getSets = sets != **null** ? "\nsets" + sets.toString() : "";

String getMaps = maps != **null** ? "\nmaps" + maps.toString() : "";

**return** getLists + getSets + getMaps;

}

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("bean.xml");

// List

CustomerFactory customList1 = (CustomerFactory) context.getBean("customerListFactoryBean1");

CustomerFactory customList2 = (CustomerFactory) context.getBean("customerListFactoryBean2");

System.***out***.println(customList1.toString());

System.***out***.println(customList2.toString());

// Set

CustomerFactory customSet1 = (CustomerFactory) context.getBean("customerSetFactoryBean1");

CustomerFactory customSet2 = (CustomerFactory) context.getBean("customerSetFactoryBean2");

System.***out***.println(customSet1.toString());

System.***out***.println(customSet2.toString());

// Map

CustomerFactory customMap1 = (CustomerFactory) context.getBean("customerMapFactoryBean1");

CustomerFactory customMap2 = (CustomerFactory) context.getBean("customerMapFactoryBean2");

System.***out***.println(customMap1.toString());

System.***out***.println(customMap2.toString());

}

}

**Factory.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:util=*"http://www.springframework.org/schema/util"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-2.0.xsd*

*http://www.springframework.org/schema/util*

*http://www.springframework.org/schema/util/spring-util-2.0.xsd"*>

<!-- ListFactoryBean -->

<!-- Way 1 -->

<bean id=*"customerListFactoryBean1"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"lists"*>

<bean class=*"org.springframework.beans.factory.config.ListFactoryBean"*>

<property name=*"targetListClass"*>

<value>java.util.ArrayList</value>

</property>

<property name=*"sourceList"*>

<list>

<value>1</value>

<value>2</value>

<value>3</value>

</list>

</property>

</bean>

</property>

</bean>

<!-- Way 2 -->

<bean id=*"customerListFactoryBean2"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"lists"*>

<util:list list-class=*"java.util.ArrayList"*>

<value>1</value>

<value>2</value>

<value>3</value>

<value>4</value>

</util:list>

</property>

</bean>

<!-- SetFactoryBean -->

<!-- Way 1 -->

<bean id=*"customerSetFactoryBean1"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"sets"*>

<bean class=*"org.springframework.beans.factory.config.SetFactoryBean"*>

<property name=*"targetSetClass"*>

<value>java.util.HashSet</value>

</property>

<property name=*"sourceSet"*>

<set>

<value>1</value>

<value>2</value>

<value>3</value>

<value>4</value>

<value>5</value>

</set>

</property>

</bean>

</property>

</bean>

<!-- Way 2 -->

<bean id=*"customerSetFactoryBean2"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"sets"*>

<util:set set-class=*"java.util.HashSet"*>

<value>1</value>

<value>2</value>

<value>3</value>

<value>4</value>

</util:set>

</property>

</bean>

<!-- MapFactoryBean -->

<!-- Way 1 -->

<bean id=*"customerMapFactoryBean1"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"maps"*>

<bean class=*"org.springframework.beans.factory.config.MapFactoryBean"*>

<property name=*"targetMapClass"*>

<value>java.util.HashMap</value>

</property>

<property name=*"sourceMap"*>

<map>

<entry key=*"1"* value=*"value 1"* />

<entry key=*"2"* value=*"value 2"* />

<entry key=*"3"* value=*"value 3"* />

</map>

</property>

</bean>

</property>

</bean>

<!-- Way 2 -->

<bean id=*"customerMapFactoryBean2"* class=*"ctcav.com.demo.factory.CustomerFactory"*>

<property name=*"maps"*>

<util:map map-class=*"java.util.HashMap"*>

<entry key=*"1"* value=*"value 1"* />

<entry key=*"2"* value=*"value 2"* />

<entry key=*"3"* value=*"value 3"* />

<entry key=*"4"* value=*"value 4"* />

</util:map>

</property>

</bean>

</beans>



1. Cấu hình kế thừa trong Spring bean

**Lớp cha là một class bình thường**

**inheritance.xml**

<bean id=*"customerInheritanceParrent"* class=*"ctcav.com.demo.inheritance.CustomerInheritance"*></bean>

<bean id=*"customerInheritance"* parent=*"customerInheritanceParrent"* init-method=*"initMethod"*>

<property name=*"action"* value=*"action 2"* />

<property name=*"type"* value=*"2"* />

</bean>



**CustomerInheritance.java**

**public** **class** CustomerInheritance {

**private** **int** type;

**private** String action;

**private** String country;

// set, get properties

**public** **void** initMethod() **throws** Exception { // Method run head

System.***out***.println("========== Init method");

}

@Override

**public** String toString() {

**return** **this**.type + " || " + **this**.action + " || " + **this**.country;

}

}

**Output**

2 || action 2 || country 1

**Lớp cha là abstract Class (Output giống với case lớp cha là một class bình thường)**

Bean kế thừa sẽ Override các property của Parrent Bean

**inheritance.xml**

<bean id=*"customerInheritanceParrent"* abstract=*"true"*>

<property name=*"type"* value=*"1"* />

<property name=*"action"* value=*"action 1"* />

<property name=*"country"* value=*"country 1"* />

</bean>

<bean id=*"customerInheritance"* parent=*"customerInheritanceParrent"* class=*"ctcav.com.demo.inheritance.CustomerInheritance"*>

<property name=*"action"* value=*"action 2"* />

<property name=*"type"* value=*"2"* />

</bean>

Ta có thể đặt class=”” ở parrent Bean hay Bean kế thừa đều như nhau

<bean id=*"customerInheritanceParrent"* class=*"ctcav.com.demo.inheritance.CustomerInheritance"* abstract=*"true"*>

<property name=*"type"* value=*"1"* />

<property name=*"action"* value=*"action 1"* />

<property name=*"country"* value=*"country 1"* />

</bean>

<bean id=*"customerInheritance"* parent=*"customerInheritanceParrent"*>

<property name=*"action"* value=*"action 2"* />

<property name=*"type"* value=*"2"* />

</bean>



1. @PostConstruct and @PreDestroy trong Spring Bean

**CustomerPostPreContructService.java**

**public** **class** CustomerPostPreContructService {

**private** String message;

// Set, get properties

@PostConstruct

**public** **void** initIt() **throws** Exception {

System.***out***.println("Init method after properties are set : " + message);

}

@PreDestroy

**public** **void** cleanUp() **throws** Exception {

System.***out***.println("Spring Container is destroy! Customer clean up");

}

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("bean.xml");

CustomerPostPreContructService cus = (CustomerPostPreContructService) context.getBean("customerPostPreContructService");

System.***out***.println(cus.getMessage());

cus.cleanUp();

}

}

**post\_pre\_contruct.xml**

Cách 1: Sử dụng Bean từ class CommonAnnotationBeanPostProcessor

<bean class=*"org.springframework.context.annotation.CommonAnnotationBeanPostProcessor"*></bean>

<bean id=*"customerPostPreContructService"* class=*"ctcav.com.demo.post\_pre\_contruct.CustomerPostPreContructService"*>

<property name=*"message"* value=*"I'm message"* />

</bean>



Cách 2: Sử dụng <context:annotation-config/>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-2.5.xsd"*>

<context:annotation-config />

<bean id=*"customerPostPreContructService"* class=*"ctcav.com.demo.post\_pre\_contruct.CustomerPostPreContructService"*>

<property name=*"message"* value=*"i'm property message"* />

</bean>

</beans>



1. Spring auto scanning components

**Cấu trúc thông thường**

**CustomerDAO.java**

**public** **class** CustomerDAO {

@Override

**public** String toString() {

**return** "Hello , This is CustomerDAO";

}

}

**CustomerService.java**

**public** **class** CustomerService {

CustomerDAO customerDAO;

**public** **void** setCustomerDAO(CustomerDAO customerDAO) {

**this**.customerDAO = customerDAO;

}

@Override

**public** String toString() {

**return** "CustomerService [customerDAO=" + customerDAO + "]";

}

}

**auto\_scanning\_components.xml**

<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-2.0.xsd"*>

<bean id=*"customerDAO"* class=*"tctav.com.demo.auto\_scanning\_components.CustomerDAO"* />

<bean id=*"customerService"* class=*"tctav.com.demo.auto\_scanning\_components.CustomerService"*>

<property name=*"customerDAO"* ref=*"customerDAO"* />

</bean>

</beans>



**Sử dụng auto scanning components**

@Component Tự động scan các component

@Repository Chỉ ra thành phần DAO trong layer này

@Service Chỉ ra thành phần Service trong layer này

@Controller Chỉ ra thành phần Controller trong layer này

Chúng ta có thể dùng @Component chung cho tất cả nhưng để rõ ràng cho từng layer thì ta nên tách chúng ra thành @Repository, @Service, @Controller.

**CustomerDAO.java**

//@Component

@Repository

**public** **class** CustomerDAO {

@Override

**public** String toString() {

**return** "Hello , This is CustomerDAO";

}

}

**CustomerService.java**

//@Component

@Service("customerService")

**public** **class** CustomerService {

CustomerDAO customerDAO;

**public** **void** setCustomerDAO(CustomerDAO customerDAO) {

**this**.customerDAO = customerDAO;

}

@Override

**public** String toString() {

**return** "CustomerService [customerDAO=" + customerDAO + "]";

}

}

**auto\_scanning\_components.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-2.5.xsd"*>

<context:component-scan base-package=*"tctav.com.demo.auto\_scanning\_components"*></context:component-scan>

</beans>



1. Spring Filter components in auto scanning

Dùng khi ta muốn chỉ định scan những component nào và không muốn scan component nào

<!-- Include -->

<context:component-scan base-package=*"tctav.com.demo.auto\_scanning\_components"*>

<context:include-filter type=*"regex"*

expression=*"tctav.com.demo.auto\_scanning\_components.\*DAO.\*"* />

<context:include-filter type=*"regex"*

expression=*"tctav.com.demo.auto\_scanning\_components.\*Service.\*"* />

</context:component-scan>

<!-- Exclude -->

<context:component-scan base-package=*"tctav.com.demo.auto\_scanning\_components"*>

<context:exclude-filter type=*"annotation"*

expression=*"org.springframework.stereotype.Service"* />

</context:component-scan>



1. Spring AutoWiring Bean

Spring Auto-Wiring Beans

<bean id=*"customer"* class=*"com.demo.main.Customer"* autowire=*"byName"* />

Có 5 mode:

* **no (default)**: Được config thông qua thuộc tính “ref”
* **byName**: Auto wiring bằng property tên. Nếu tên của bean giống với tên của property bean sẽ được auto wire
* **byType**: Auto wiring bằng data type. Nếu data type của bean giống với data type của property bean sẽ được auto wire
* **constructor**: by type trong argument constructor
* **autodetect**: mặc định sẽ autowired constructor. Nếu không có sẽ autowire by type

1. No

**public** **class** Customer {

**private** Person person;

**public** **void** setPerson(Person person) {

**this**.person = person;

}

}

<bean id=*"person"* class=*"com.demo.main.Person"* />

<bean id=*"customer"* class=*"com.demo.main.Customer"*>

<property name=*"person"* ref=*"person"* />

</bean>

2. byName

**public** **class** Agent {

**private** Address address;

**public** **void** setAddress(Address address) {

**this**.address = address;

}

}

**public** **class** Address {

**private** String fulladdress;

**public** **void** setFulladdress(String fulladdress) {

**this**.fulladdress = fulladdress;

}

}

<bean id=*"agent"* class=*"com.demo.main.Agent"* autowire=*"byName"* />

<bean id=*"address"* class=*"com.demo.main.Address"* >

<property name=*"fulladdress"* value=*"dfsd fafdf sfsadf"* />

</bean>

3. byType

**public** **class** Person {

**private** Ability ability;

**public** **void** setAbility(Ability ability) {

**this**.ability = ability;

}

}

**public** **class** Ability {

**private** String skill;

**public** **void** setSkill(String skill) {

**this**.skill = skill;

}

}

<bean id=*"personByType"* class=*"com.demo.main.Person"* autowire=*"byType"* />

<bean id=*"invisible"* class=*"com.demo.main.Ability"* >

<property name=*"skill"* value=*"Invisible"* />

</bean>

4. constructor

**public** **class** Developer {

**private** Language language;

// autowire by constructor

**public** Developer(Language language) {

**this**.language = language;

}

}

**public** **class** Language {

**private** String name;

**public** **void** setName(String name) {

**this**.name = name;

}

}

<bean id=*"developer"* class=*"com.demo.main.Developer"* autowire=*"constructor"* />

<bean id=*"language"* class=*"com.demo.main.Language"* >

<property name=*"name"* value=*"Java"* />

</bean>

5. autodetect

Mặc định sẽ autowired constructor. Nếu không có sẽ autowire by type