Spring的依赖

与Spring的继承类似，依赖也是bean与bean之间的一种关联方式，配置依赖关系之后，被依赖的bean一定优先创建，再创建依赖的bean，A依赖于B，先创建B，再创建A。

IoC容器创建bean的顺序是由spring.xml中的顺序来决定的，从上向下执行创建。

```
<!-- Teacher -->
<bean id="teacher" class="com.southwind.entity.Teacher" depends-
on="student"></bean>

<!-- Student -->
<bean id="student" class="com.southwind.entity.Student"></bean>
```

Spring的p命名空间

p命名空间的作用是简化spring.xml中的bean的配置，包括属性复制，依赖注入（DI）。

```
<bean id="teacher2" class="com.southwind.entity.Teacher" p:id="3"
p:name="王五"></bean>
<bean id="student2" class="com.southwind.entity.Student" p:id="4"
p:name="小明"></bean>
<bean id="clazz" class="com.southwind.entity.Clazz" p:id="1" p:name="一班"
p:teacher-ref="teacher"></bean>
```

Spring IoC工厂方法创建对象

IoC是典型的工厂模式，IoC通过工厂模式创建bean有两种方式：

- 静态工厂方法

Car

```
package com.southwind.entity;

public class Car {
    private int num;
    private String brand;

    public int getNum() {
        return num;
    }
```

```
    public void setNum(int num) {
        this.num = num;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    @Override
    public String toString() {
        return "Car{" +
                "num=" + num +
                ", brand='" + brand + '\'' +
                '}';
    }

    public Car(int num, String brand) {
        this.num = num;
        this.brand = brand;
    }
}
```

StaticCarFactory

```
package com.southwind.factory;

import com.southwind.entity.Car;

import java.util.HashMap;
import java.util.Map;

public class StaticCarFactory {
    private static Map<Integer, Car> carMap;
    static {
        carMap = new HashMap<>();
        carMap.put(1,new Car(1,"奥迪"));
        carMap.put(2,new Car(2,"宝马"));
    }

    public static Car getCar(int num){
        return carMap.get(num);
    }
}
```

spring.xml

```
<!-- 配置静态工厂创建Car对象 -->
<bean id="car" class="com.southwind.factory.StaticCarFactory" factory-
method="getCar">
    <constructor-arg value="2"></constructor-arg>
</bean>
```

- 实例工厂方法

InstanceCarFactory

```
package com.southwind.factory;

import com.southwind.entity.Car;

import java.util.HashMap;
import java.util.Map;

public class InstanceCarFactory {
    private Map<Integer, Car> carMap;
    public InstanceCarFactory(){
        carMap = new HashMap<>();
        carMap.put(1,new Car(1,"奔驰"));
        carMap.put(2,new Car(2,"宝马"));
    }

    public Car getCar(int num){
        return carMap.get(num);
    }
}
```

spring.xml

```
<!-- 配置实例工厂对象 -->
<bean id="instanceCarFactory"
class="com.southwind.factory.InstanceCarFactory"></bean>

<!-- 通过实例工厂对象获取Car对象 -->
<bean id="car" factory-bean="instanceCarFactory" factory-method="getCar" >
    <constructor-arg value="1"></constructor-arg>
</bean>
```

对比两种方式的区别，静态工厂方法的方式获取Car对象，不需要实例化工厂对象，因为静态工厂的静态方法不需要创建对象即可调用，所以spring.xml中只需要配置一个Car bean，不需要配置Factory bean。

实例工厂方法获取Car对象，必须先实例化工厂对象，因为调用的是非静态（实例）方法，必须通过实例化对象来调用方法，不能之间通过类来调用，所以spring.xml中需要先配置Factory bean，再配置Car bean。

IoC的自动装载（autowire）

IocDI可以通过配置property的ref属性将bean进行依赖注入，同时Spring还提供了另外一种更加简便的方式：自动装载，不需要手动配置property，IoC容器会自动选择bean完成依赖注入。

自动装载有两种方式：

- byName：通过属性名完成自动装载。

```
<!-- 配置Car -->
<bean id="car" class="com.southwind.factory.StaticCarFactory" factory-
method="getCar">
    <constructor-arg value="1"></constructor-arg>
</bean>

<!-- 配置Person -->
<bean id="person" class="com.southwind.entity.Person" autowire="byName">
    <property name="id" value="1"></property>
    <property name="name" value="张三"></property>
</bean>
```

- byType：通过属性的数据类型完成自动装载。

```
<!-- 配置Car -->
<bean id="car2" class="com.southwind.factory.StaticCarFactory" factory-
method="getCar">
    <constructor-arg value="1"></constructor-arg>
</bean>

<!-- 配置Person -->
<bean id="person" class="com.southwind.entity.Person" autowire="byType">
    <property name="id" value="1"></property>
    <property name="name" value="张三"></property>
</bean>
```

如果IoC容器同时存在多个数据类型相同的bean，会抛出异常。