08 Spring_配置详解_自动注入&&候选者

`Spring`

一：自动注入(扩展)_4种

> 1、可分别在
> |-全局：

```
<beans default-autowire="?" ...>
```

> |-局部：

```
<bean autowire="?" ...>
```

> 2、自动注入会大量减少相关配置，但同时也导致通过配置文件无法看到bean之间的依赖关系

1、no(默认值)

> 1、不进行自动注入，需要手动的使用ref进行注入

2、byName

> 1、通过bean名称进行setter自动注入；
> 2、如bean有setter方法setUserDao,则从容器中需找一个userDao的bean进行注入，如果未找到或找到多个则异常；
> 3、实体类

```
public class Users {
      private  Integer  id;
      private  String name;
      private  Car  car;

        public void setCar(Car car) {
            this.car = car;
        }
        //其他对应的get set 方法
        }
```

4、配置文件

```
<bean name="car" class="com.spring.bean.Car">
        <property name="name" value="奔驰5"></property>
        <property name="clor" value="红色5"></property>
    </bean>
    <!--构造方法 自动注入-Constructor  -->
    <bean name="users,user1" class="com.spring.bean.Users"  autowire="byName">
      <property name="name">
        <value>小张</value>
      </property>
    </bean>
```

5、测试

```
    @org.junit.Test
    public  void getVoid01(){
        //1.创建容器对象
    ClassPathXmlApplicationContext classPath= new ClassPathXmlApplicationContext("applicationContext.xml");
        //2.找容器要对象
    Users bean = (Users)classPath.getBean("user1");
        System.out.println("bean:"+bean);
    }
```

3、byType(常用)

> 0、测试同上
> 1、通过bean类型进行setter自动注入，如果bean有setter方法 setUserDao(UserDao)；
> 2、则从容器中寻找bean<类型兼容>UserDao类型的bean进行注入，如果未找到或找到多个则异常；
> 3、注:类型兼容是指接口类型的实现类或父类的继承子类
> 4、实体类

```
public class Users {
      private  Integer  id;
      private  String name;

      private  Car  car;
     //对应的get set 方法
```

> 5、配置文件

```
    <bean name="car" class="com.spring.bean.Car">
        <property name="name" value="奔驰2"></property>
        <property name="clor" value="红色2"></property>
    </bean>
    <!--构造方法 自动注入-Constructor  -->
    <bean name="byType" class="com.spring.bean.Users"  autowire="byType
"></bean>
```

4、Constructor

> 1、类似于byType，但针对的是构造参数注入，如果未找到或多个则异常
> 2、实体类

```
public class Users {
      private  Integer  id;
      private  String name;

      private  Car  car;
      /**
```

```
         * 必须要有对应的构造方法
         * 方可进行自动注入
         */
          public Users( Car car) {
          System.out.println("自动  构造函数的注入");
          this.car = car;
      }
      //对应的get set 方法
```

> 3、配置文件

```
      <bean name="car" class="com.spring.bean.Car">
          <property name="name" value="奔驰1"></property>
          <property name="clor" value="红色1"></property>
      </bean>
      <!--构造方法 自动注入-Constructor  -->
      <bean name="constructor" class="com.spring.bean.Users"  autowire="c
onstructor"></bean>
```

二：自动注入和手动注入混合

> 1、如果自动和手动注入混合使用
> 2、则先进行自动注入，再进行手动注入
> 3、手动注入的值覆盖自动注入的值

三：多个候选者的处理

> 1、通过byType和constructor两种方式可能会出现多个bean满足注入条件，我们称这些bean为候选者;

1、候选者识别

1-1、全局指定

> 1、在根标签< beans>上有一个属性default-autowire-candidates.

> 2、可用于指定容器中哪些bean作为默认的自动注入候选者(可使用逗号分割，写多个).
> 3、default-autowire-candidates默认值为空，则表示所有bean都为候选者

```
<beans  default-autowire-candidates="car,car1"  .....>
```

1-2、局部指定

> 1、在< bean>上使用autowire-candidate="true"来指明此bean要作为候选者，覆盖全局设置。

```
<bean name="car" class="com.spring.bean.Car"  autowire-candidate="false">
        <property name="name" value="奔驰5"></property>
        <property name="clor" value="红色5"></property>
    </bean>
    <bean name="car1" class="com.spring.bean.Car" autowire-candidate="true">
        <property name="name" value="奔驰1"></property>
        <property name="clor" value="红色1"></property>
    </bean>
    <!--构造方法 自动注入-Constructor  -->
    <bean name="users,user1" class="com.spring.bean.Users"  autowire="constructor">
      <property name="name">
        <value>小张</value>
      </property>
    </bean>
```

2、主要候选者

> 1、如果还有多个候选者，还可在< bean>上定义`primary= “true"来表示其为主要的候选者。

```
    <bean name="car" class="com.spring.bean.Car"  autowire-candidate="true" primary="true">
        <property name="name" value="奔驰5"></property>
        <property name="clor" value="红色5"></property>
    </bean>
```

```
    <bean name="car1" class="com.spring.bean.Car" autowire-candidate="true">
        <property name="name" value="奔驰1"></property>
        <property name="clor" value="红色1"></property>
    </bean>
```