控制反转IOC

2021年9月19日 18:10

控制反转是Spring容器的一个重要特征

控制反转，指的是在Spring容器中，Bean对象之间的依赖不由它们自己来管理，而是由Spring容器负责管理对象之间的依赖

依赖注入DI

Spring容器通过依赖注入（DI）的方式来实现控制反转
我们需要通过注解的方式（当然，也可以用配置文件、java代码）来告诉Spring容器这些对象之间是会相互关联的，我们最常用的注解是**@AutoWired**
这个注解可以放在Bean对象的属性前面、Bean对象的方法之前，来告诉Spring容器，这样的一个属性或者方法的参数是需要由Spring容器来注入对象

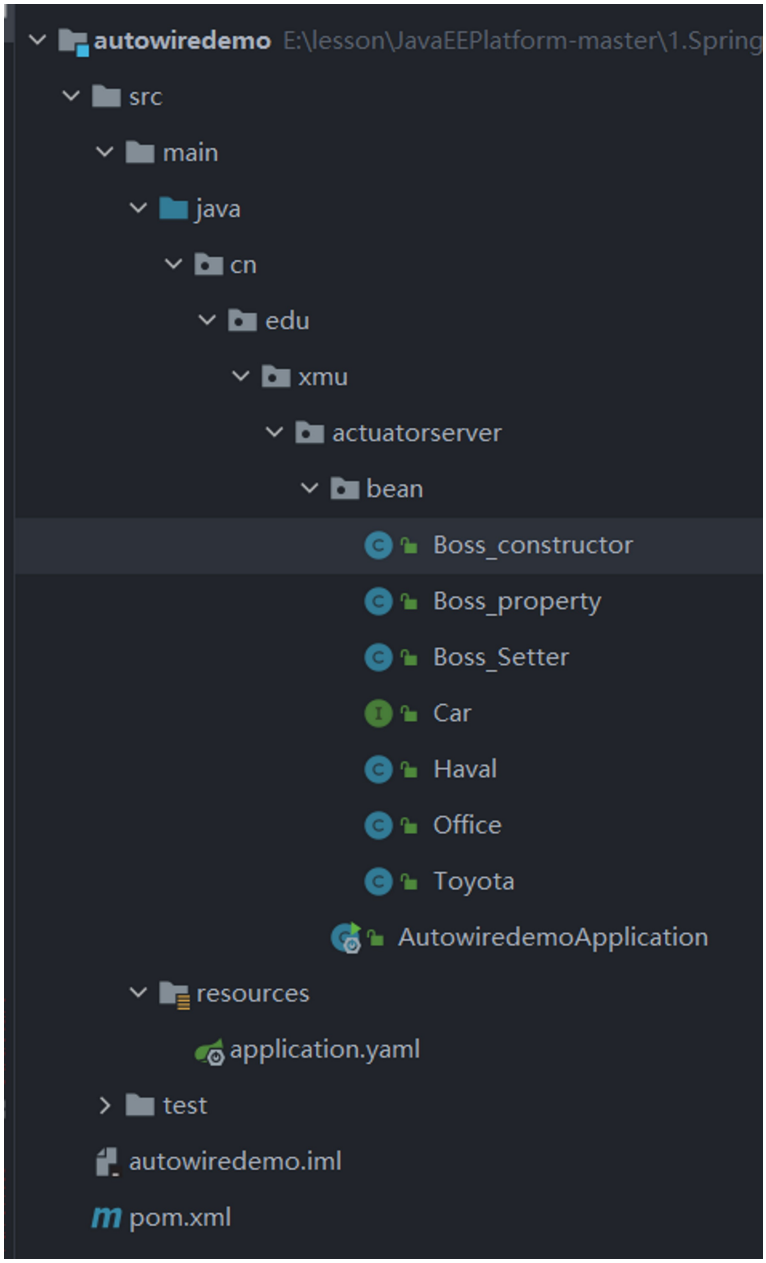
默认匹配规则

Spring容器会去查询同一类型的Bean对象

- 如果**同一类型的Bean对象只有一个**，就会把Bean对象装配给对应的属性，或者装配给方法的对应参数
- 如果**同一类型的Bean对象不止一个**，Spring容器会根据**变量的名称**来查找，即以变量的名称（或方法参数的名称）作为id来查找对应的Bean对象

实例解析

文件结构



我们定义了三个Bean对象：Boss、Car、Office，Car有两个子类Haval和Toyota，Office是一个空类

用Boss类来测试@Autowired的三种注入方式：注入构造函数、注入一般方法、注入属性

注入构造函数

```java
@Component
public class Boss_constructor {
    private Car car;
    private Office office;

    @Autowired
    public Boss_constructor(Car toyota, Office office){
        this.car = toyota;
        this.office = office;
    }
}
```

可以注意到，Car类型的Bean对象有两个，toyota和haval，所以当Spring容器按照Car类型去查找时，结果不止一个，所以会按照参数去查找对象，在这里就是用小写的toyota去查找。我们定义Toyota类时没有指定id，因此Toyota类的Bean对象id就是类名首字母小写toyota，Spring容器就是按照这个规则找到了对应的Bean对象

注入一般方法

```java
@Component("boss_Setter")
public class Boss_Setter {

    private Car car;
    private Office office;

    @Autowired
    public void setCar(Car toyota) { this.car = toyota; }

    @Autowired
    public void setOffice(Office office) { this.office = office; }

}
```

注入属性

```java
@Component
public class Boss_property {
    @Autowired
    private Car haval;
    @Autowired
    private Office office;
}
```

注入结果

可以通过后台监控，看到Bean自动注入的结果

boss_Setter

cn.edu.xmu.actuatorserver.bean.Boss_Setter

资源	file [E:\lesson\JavaEEPlatform-master\Boss_Setter.class]
依赖关系	office
	toyota

boss_constructor

cn.edu.xmu.actuatorserver.bean.Boss_constructor

资源	file [E:\lesson\JavaEEPlatform-maste an\Boss_constructor.class]
依赖关系	toyota
	office

boss_property

cn.edu.xmu.actuatorserver.bean.Boss_property

资源	file [E:\lesson\JavaEEPlatform-ma an\Boss_property.class]
依赖关系	haval
	office