03_1-Spring IOC的注解使用

03SpringIOC的注解应用

在之前的项目中，我们都是通过xml文件进行bean或者某些属性的赋值，其实还有另外一种注解的方式，在企业开发中使用的很多，在bean上添加注解，可以快速的将bean注册到ioc容器。

1、使用注解的方式注册bean到IOC容器中

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

    <!--
    如果想要将自定义的bean对象添加到IOC容器中，需要在类上添加某些注解
    Spring中包含4个主要的组件添加注解：
    @Controller:控制器，推荐给controller层添加此注解
    @Service:业务逻辑，推荐给业务逻辑层添加此注解
    @Repository:仓库管理，推荐给数据访问层添加此注解
    @Component:给不属于以上基层的组件添加此注解
    注意：我们虽然人为的给不同的层添加不同的注解，但是在spring看来，可以在任意层添加任意注解
            spring底层是不会给具体的层次验证注解，这样写的目的只是为了提高可读性，最偷懒的方式
            就是给所有想交由IOC容器管理的bean对象添加component注解

    使用注解需要如下步骤：
```

```
    1、添加上述四个注解中的任意一个
    2、添加自动扫描注解的组件，此操作需要依赖context命名空间
    3、添加自动扫描的标签context:component-scan

   注意：当使用注解注册组件和使用配置文件注册组件是一样的，但是要注意：
            1、组件的id默认就是组件的类名首字符小写，如果非要改名字的话，直接在注解中添加即可
            2、组件默认情况下都是单例的,如果需要配置多例模式的话，可以在注解下添加@Scope注解
    -->
    <!--
    定义自动扫描的基础包:
    base-package:指定扫描的基础包，spring在启动的时候会将基础包及子包下所有加了注解的类都自动
                扫描进IOC容器
    -->
    <context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>
</beans>
```

PersonController.java

```java
package cn.tulingxueyuan.controller;

import org.springframework.stereotype.Controller;

@Controller
public class PersonController {
    public PersonController() {
        System.out.println("创建对象");
    }
}
```

PersonService.java

```java
package cn.tulingxueyuan.service;

import org.springframework.stereotype.Service;

@Service
public class PersonService {
}
```

PersonDao.java

```java
package cn.tulingxueyuan.dao;

import org.springframework.stereotype.Repository;

@Repository("personDao")
@Scope(value="prototype")
public class PersonDao {
}
```

2、定义扫描包时要包含的类和不要包含的类

当定义好基础的扫描包后，在某些情况下可能要有选择性的配置是否要注册bean到IOC容器中，此时可以通过如下的方式进行配置。

applicationContext.xml

```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.xsd
        http://www.springframework.org/schema/context
```

```
        http://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="cn.tulingxueyuan" use-default-filters="false">
        <!--
        当定义好基础扫描的包之后，可以排除包中的某些类，使用如下的方式：
        type:表示指定过滤的规则
            annotation：按照注解进行排除，标注了指定注解的组件不要,expression表示要过滤的注解
            assignable：指定排除某个具体的类，按照类排除，expression表示不注册的具体类名
            aspectj：后面讲aop的时候说明要使用的aspectj表达式，不用
            custom：定义一个typeFilter,自己写代码决定哪些类被过滤掉，不用
            regex：使用正则表达式过滤，不用
        -->
<!--        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>-->

        <!--指定只扫描哪些组件，默认情况下是全部扫描的，所以此时要配置的话需要在component-scan标签中添加 use-default-filters="false"-->
        <context:include-filter type="assignable" expression="cn.tulingxueyuan.service.PersonService"/>
    </context:component-scan>
</beans>
```

3、使用@AutoWired进行自动注入

使用注解的方式实现自动注入需要使用@AutoWired注解。

PersonController.java

```
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.service.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

@Controller
public class PersonController {
```

```
9
10      @Autowired
11      private PersonService personService;
12
13      public PersonController() {
14          System.out.println("创建对象");
15      }
16
17      public void getPerson(){
18          personService.getPerson();
19      }
20  }
```

PersonService.java

```
1   package cn.tulingxueyuan.service;
2
3   import cn.tulingxueyuan.dao.PersonDao;
4   import org.springframework.beans.factory.annotation.Autowired;
5   import org.springframework.stereotype.Service;
6
7   @Service
8   public class PersonService {
9
10      @Autowired
11      private PersonDao personDao;
12
13      public void getPerson(){
14          personDao.getPerson();
15      }
16  }
```

PersonDao.java

```java
package cn.tulingxueyuan.dao;

        import org.springframework.stereotype.Repository;

@Repository
public class PersonDao {

    public void getPerson(){
        System.out.println("PersonDao:getPerson");
    }
}
```

注意：当使用AutoWired注解的时候，自动装配的时候是根据类型实现的。

1、如果只找到一个，则直接进行赋值，

2、如果没有找到，则直接抛出异常，

3、如果找到多个，那么会按照变量名作为id继续匹配，

1、匹配上直接进行装配

2、如果匹配不上则直接报异常

PersonServiceExt.java

```java
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.dao.PersonDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
```

```java
public class PersonServiceExt extends PersonService{

    @Autowired
    private PersonDao personDao;

    public void getPerson(){
        System.out.println("PersonServiceExt......");
        personDao.getPerson();
    }
}
```

PersonController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.service.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

@Controller
public class PersonController {

    @Autowired
    private PersonService personServiceExt;

    public PersonController() {
        System.out.println("创建对象");
    }

    public void getPerson(){
        personServiceExt.getPerson();
```

```
    }
}
```

还可以使用@Qualifier注解来指定id的名称，让spring不要使用变量名,当使用@Qualifier注解的时候也会有两种情况：

2、找不到，就会报错

PersonController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.service.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;

@Controller
public class PersonController {

    @Autowired
    @Qualifier("personService") 再指定id，两个配合使用
    private PersonService personServiceExt2;

    public PersonController() {
        System.out.println("创建对象");
    }

    public void getPerson(){
        personServiceExt2.getPerson();
    }
}
```

通过上述的代码我们能够发现，使用@AutoWired肯定是能够装配上的，如果装配不上就会报错。

4、@AutoWired可以进行定义在方法上

当我们查看@AutoWired注解的源码的时候发现，此注解不仅可以使用在成员变量上，也可以使用在方法上。

PersonController.java

```
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.dao.PersonDao;
import cn.tulingxueyuan.service.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;

@Controller
public class PersonController {

    @Qualifier("personService")
    @Autowired
    private PersonService personServiceExt2;

    public PersonController() {
        System.out.println("创建对象");
    }

    public void getPerson(){
        System.out.println("personController..."+personServiceExt2);
//        personServiceExt2.getPerson();
    }

     /**
```

```
     * 当方法上有@AutoWired注解时：
     *  1、此方法在bean创建的时候会自动调用
     *  2、这个方法的每一个参数都会自动注入值
     * @param personDao
     */
    @Autowired
    public void test(PersonDao personDao){
        System.out.println("此方法被调用:"+personDao);
    }

    /**
     * @Qualifier注解也可以作用在属性上，用来被当作id去匹配容器中的对象，如果没有
     * 此注解，那么直接按照类型进行匹配
     * @param personService
     */
    @Autowired
    public void test2(@Qualifier("personServiceExt") PersonService personService){
        System.out.println("此方法被调用: "+personService);
    }
}
```

5、自动装配的注解@AutoWired，@Resource

在使用自动装配的时候，出了可以使用@AutoWired注解之外，还可以使用@Resource注解，大家需要知道这两个注解的区别。

1、@AutoWired:是spring中提供的注解，@Resource:是jdk中定义的注解，依靠的是java的标准

2、@AutoWired默认是按照类型进行装配，默认情况下要求依赖的对象必须存在，@Resource默认是按照名字进行匹配的，同时可以指定name属性。

3、@AutoWired只适合spring框架，而@Resource扩展性更好

PersonController.java

```
package cn.tulingxueyuan.controller;
```

```java

import cn.tulingxueyuan.dao.PersonDao;
import cn.tulingxueyuan.service.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;

import javax.annotation.Resource;

@Controller
public class PersonController {

    @Qualifier("personService")
    @Resource
    private PersonService personServiceExt2;

    public PersonController() {
        System.out.println("创建对象");
    }

    public void getPerson(){
        System.out.println("personController..."+personServiceExt2);
        personServiceExt2.getPerson();
    }

    /**
     * 当方法上有@AutoWired注解时：
     *  1、此方法在bean创建的时候会自动调用
     *  2、这个方法的每一个参数都会自动注入值
     * @param personDao
```

```
     */
    @Autowired
    public void test(PersonDao personDao){
        System.out.println("此方法被调用:"+personDao);
    }

    /**
     * @Qualifier注解也可以作用在属性上，用来被当作id去匹配容器中的对象，如果没有
     * 此注解，那么直接按照类型进行匹配
     * @param personService
     */
    @Autowired
    public void test2(@Qualifier("personServiceExt") PersonService personService){
        System.out.println("此方法被调用: "+personService);
    }
}
```

1、@Autowired是spring自带的， @Resource是JSR250规范实现的，需要导入不同的包

3、~~@Autowired~~ @Resource是按照名称匹配的

4、@Autowired如果需要按照名称匹配需要和@Qualifier一起使用

6、泛型依赖注入

为了讲解泛型依赖注入，首先我们需要先写一个基本的案例，按照我们之前学习的知识：

Student.java

```
package cn.tulingxueyuan.bean;

public class Student {
}
```

Teacher.java

```java
package cn.tulingxueyuan.bean;

public class Teacher {
}
```

BaseDao.java

```java
package cn.tulingxueyuan.dao;

import org.springframework.stereotype.Repository;

@Repository
public abstract class BaseDao<T> {

    public abstract void save();
}
```

StudentDao.java

```java
package cn.tulingxueyuan.dao;

import cn.tulingxueyuan.bean.Student;
import org.springframework.stereotype.Repository;

@Repository
public class StudentDao extends BaseDao<Student>{
    public void save() {
        System.out.println("保存学生");
    }
```

```
}
```

TeacherDao.java

```
package cn.tulingxueyuan.dao;

import cn.tulingxueyuan.bean.Teacher;
import org.springframework.stereotype.Repository;

@Repository
public class TeacherDao extends BaseDao<Teacher> {
    public void save() {
        System.out.println("保存老师");
    }
}
```

StudentService.java

```
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.dao.StudentDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class StudentService {

    @Autowired
    private StudentDao studentDao;

    public void save(){
```

```
        studentDao.save();
    }
}
```

TeacherService.java

```
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.dao.TeacherDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class TeacherService {
    @Autowired
    private TeacherDao teacherDao;

    public void save(){
        teacherDao.save();
    }
}
```

MyTest.java

```
import cn.tulingxueyuan.service.StudentService;
import cn.tulingxueyuan.service.TeacherService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import javax.sql.DataSource;
```

```
import java.sql.SQLException;

public class MyTest {
    public static void main(String[] args) throws SQLException {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        StudentService studentService = context.getBean("studentService",StudentService.class);
        studentService.save();

        TeacherService teacherService = context.getBean("teacherService",TeacherService.class);
        teacherService.save();
    }
}
```

上述代码是我们之前的可以完成的功能，但是可以思考，Service层的代码是否能够改写：

BaseService.java

```
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.dao.BaseDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

public class BaseService<T> {

    @Autowired
    BaseDao<T> baseDao;

    public void save(){
        System.out.println("自动注入的对象："+baseDao);
        baseDao.save();
```

```
    }
}
```

StudentService.java

```
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.bean.Student;
import cn.tulingxueyuan.dao.StudentDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class StudentService extends BaseService<Student> {

}
```

TeacherService.java

```
package cn.tulingxueyuan.service;

import cn.tulingxueyuan.bean.Teacher;
import cn.tulingxueyuan.dao.TeacherDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class TeacherService extends BaseService<Teacher>{

}
```

- 怎样开启注解装配?
- @Component, @Controller, @Repository, @Service 有何区别?
- 当使用@AutoWired匹配到多个类型怎么解决?