

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

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:context="http://www.springframework.org/schema/context"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6      http://www.springframework.org/schema/beans/spring-beans.xsd
7      http://www.springframework.org/schema/context
8      http://www.springframework.org/schema/context/spring-context.xsd">
9
10     <!--
11     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
12     Spring中包含4个主要的组件添加注解：
13     @Controller:控制器，推荐给controller层添加此注解
14     @Service:业务逻辑，推荐给业务逻辑层添加此注解
15     @Repository:仓库管理，推荐给数据访问层添加此注解
16     @Component:给不属于以上基层的组件添加此注解
17     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
18           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
19           就是给所有想交由IOC容器管理的bean对象添加component注解
20
21     使用注解需要如下步骤：
22     1、添加上述四个注解中的任意一个
23     2、添加自动扫描注解的组件，此操作需要依赖context命名空间
24     3、添加自动扫描的标签context:component-scan
25
26     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：
27           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
28           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
29     -->
30     <!--
31     定义自动扫描的基础包：
32     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
33           扫描进IOC容器
34     -->
```

```
35     <context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>
36 </beans>
```

PersonController.java

```
1 package cn.tulingxueyuan.controller;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class PersonController {
7     public PersonController() {
8         System.out.println("创建对象");
9     }
10 }
```

PersonService.java

```
1 package cn.tulingxueyuan.service;
2
3 import org.springframework.stereotype.Service;
4
5 @Service
6 public class PersonService {
7 }
```

PersonDao.java

```
1 package cn.tulingxueyuan.dao;
2
3 import org.springframework.stereotype.Repository;
4
5 @Repository("personDao")
6 @Scope(value="prototype")
7 public class PersonDao {
8 }
```

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6       http://www.springframework.org/schema/beans/spring-beans.xsd
7       http://www.springframework.org/schema/context
8       http://www.springframework.org/schema/context/spring-context.xsd">
9   <context:component-scan base-package="cn.tulingxueyuan" use-default-filters="false">
10       <!--
11           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：
12           type:表示指定过滤的规则
13               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
14               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
15               aspectj: 后面讲aop的时候说明要使用的aspectj表达式，不用
16               custom: 定义一个typeFilter，自己写代码决定哪些类被过滤掉，不用
17               regex: 使用正则表达式过滤，不用
18       -->
19   <!--           <context:exclude-filter type="annotation" expression="org.springframework.ste
20
21       <!--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
22       <context:include-filter type="assignable" expression="cn.tulingxueyuan.service.Pe
23   </context:component-scan>
24 </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

```
1 package cn.tulingxueyuan.controller;
2
3 import cn.tulingxueyuan.service.PersonService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Controller;
6
7 @Controller
8 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

```

1  package cn.tulingxueyuan.dao;
2
3      import org.springframework.stereotype.Repository;
4
5  @Repository
6  public class PersonDao {

```

```

7
8     public void getPerson(){
9         System.out.println("PersonDao:getPerson");
10    }
11 }

```

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

```

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 PersonServiceExt extends PersonService{
9
10     @Autowired
11     private PersonDao personDao;
12
13     public void getPerson(){
14         System.out.println("PersonServiceExt.....");
15         personDao.getPerson();
16     }
17 }

```

PersonController.java

```

1  package cn.tulingxueyuan.controller;
2
3  import cn.tulingxueyuan.service.PersonService;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.stereotype.Controller;

```

```

6
7 @Controller
8 public class PersonController {
9
10     @Autowired
11     private PersonService personServiceExt;
12
13     public PersonController() {
14         System.out.println("创建对象");
15     }
16
17     public void getPerson(){
18         personServiceExt.getPerson();
19     }
20 }

```

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

```

1 package cn.tulingxueyuan.controller;
2
3 import cn.tulingxueyuan.service.PersonService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.beans.factory.annotation.Qualifier;
6 import org.springframework.stereotype.Controller;
7
8 @Controller
9 public class PersonController {
10
11     @Autowired
12     @Qualifier("personService")
13     private PersonService personServiceExt2;
14
15     public PersonController() {
16         System.out.println("创建对象");
17     }
18

```

```
19     public void getPerson(){
20         personServiceExt2.getPerson();
21     }
22 }
```

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的成员变量上，也可以使用在方法上。

PersonController.java

```
1  package cn.tulingxueyuan.controller;
2
3  import cn.tulingxueyuan.dao.PersonDao;
4  import cn.tulingxueyuan.service.PersonService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.beans.factory.annotation.Qualifier;
7  import org.springframework.stereotype.Controller;
8
9  @Controller
10 public class PersonController {
11
12     @Qualifier("personService")
13     @Autowired
14     private PersonService personServiceExt2;
15
16     public PersonController() {
17         System.out.println("创建对象");
18     }
19
20     public void getPerson(){
21         System.out.println("personController..." + personServiceExt2);
22         // personServiceExt2.getPerson();
23     }
24
25     /**
26      * 当方法上有@Autowired注解时：
27      * 1、此方法在bean创建的时候会自动调用
28      * 2、这个方法的每一个参数都会自动注入值
```

```

29     * @param personDao
30     */
31     @Autowired
32     public void test(PersonDao personDao){
33         System.out.println("此方法被调用:"+personDao);
34     }
35
36     /**
37     * @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
38     * 此注解，那么直接按照类型进行匹配
39     * @param personService
40     */
41     @Autowired
42     public void test2(@Qualifier("personServiceExt") PersonService personService){
43         System.out.println("此方法被调用: "+personService);
44     }
45 }

```

5、自动装配的注解@AutoWired, @Resource

在使用自动装配的时候，除了可以使用@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

```

1  package cn.tulingxueyuan.controller;
2
3  import cn.tulingxueyuan.dao.PersonDao;
4  import cn.tulingxueyuan.service.PersonService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.beans.factory.annotation.Qualifier;
7  import org.springframework.stereotype.Controller;
8
9  import javax.annotation.Resource;
10
11  @Controller
12  public class PersonController {

```



```

13
14     @Qualifier("personService")
15     @Resource
16     private PersonService personServiceExt2;
17
18     public PersonController() {
19         System.out.println("创建对象");
20     }
21
22     public void getPerson(){
23         System.out.println("personController..." + personServiceExt2);
24         personServiceExt2.getPerson();
25     }
26
27     /**
28      * 当方法上有@Autowired注解时:
29      * 1、此方法在bean创建的时候会自动调用
30      * 2、这个方法的每一个参数都会自动注入值
31      * @param personDao
32      */
33     @Autowired
34     public void test(PersonDao personDao){
35         System.out.println("此方法被调用:" + personDao);
36     }
37
38     /**
39      * @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
40      * 此注解，那么直接按照类型进行匹配
41      * @param personService
42      */
43     @Autowired
44     public void test2(@Qualifier("personServiceExt") PersonService personService){
45         System.out.println("此方法被调用: " + personService);
46     }
47 }

```

- 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

```
1 package cn.tulingxueyuan.bean;
2
3 public class Student {
4 }
```

Teacher.java

```
1 package cn.tulingxueyuan.bean;
2
3 public class Teacher {
4 }
```

BaseDao.java

```
1 package cn.tulingxueyuan.dao;
2
3 import org.springframework.stereotype.Repository;
4
5 @Repository
6 public abstract class BaseDao<T> {
7
8     public abstract void save();
9 }
```

StudentDao.java

```
1 package cn.tulingxueyuan.dao;
2
3 import cn.tulingxueyuan.bean.Student;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7 public class StudentDao extends BaseDao<Student>{
8     public void save() {
9         System.out.println("保存学生");
10    }
```

```
10     }  
11 }
```

TeacherDao.java

```
1 package cn.tulingxueyuan.dao;  
2  
3 import cn.tulingxueyuan.bean.Teacher;  
4 import org.springframework.stereotype.Repository;  
5  
6 @Repository  
7 public class TeacherDao extends BaseDao<Teacher> {  
8     public void save() {  
9         System.out.println("保存老师");  
10    }  
11 }
```

StudentService.java

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

TeacherService.java

```
1 package cn.tulingxueyuan.service;
```

```

2
3 import cn.tulingxueyuan.dao.TeacherDao;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6
7 @Service
8 public class TeacherService {
9     @Autowired
10    private TeacherDao teacherDao;
11
12    public void save(){
13        teacherDao.save();
14    }
15 }

```

MyTest.java

```

1 import cn.tulingxueyuan.service.StudentService;
2 import cn.tulingxueyuan.service.TeacherService;
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 import javax.sql.DataSource;
7 import java.sql.SQLException;
8
9 public class MyTest {
10    public static void main(String[] args) throws SQLException {
11        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
12        StudentService studentService = context.getBean("studentService", StudentService.class);
13        studentService.save();
14
15        TeacherService teacherService = context.getBean("teacherService", TeacherService.class);
16        teacherService.save();
17    }
18 }

```

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

```

1 package cn.tulingxueyuan.service;

```

```

2
3 import cn.tulingxueyuan.dao.BaseDao;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6
7 public class BaseService<T> {
8
9     @Autowired
10     BaseDao<T> baseDao;
11
12     public void save(){
13         System.out.println("自动注入的对象: "+baseDao);
14         baseDao.save();
15     }
16 }

```

StudentService.java

```

1 package cn.tulingxueyuan.service;
2
3 import cn.tulingxueyuan.bean.Student;
4 import cn.tulingxueyuan.dao.StudentDao;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class StudentService extends BaseService<Student> {
10
11 }

```

TeacherService.java

```

1 package cn.tulingxueyuan.service;
2
3 import cn.tulingxueyuan.bean.Teacher;
4 import cn.tulingxueyuan.dao.TeacherDao;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service

```

```
9 public class TeacherService extends BaseService<Teacher>{  
10  
11 }
```

- 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?