# Spring 注解驱动开发

# 一、前言

spring就是想尽办法将new做的更简单，更完美，更可配置。

Spring的一个核心功能是IOC，就是将Bean初始化加载到容器中，Bean是如何加载到容器的，可以使用Spring注解方式或者Spring XML配置方式。

Spring注解方式减少了配置文件内容，更加便于管理，并且使用注解可以大大提高了开发效率！

注解本身是没有功能的，和xml一样，注解和xml都是一种元数据，元数据即解释数据的数据，也就是所谓的配置。

**1、xml和注解的最佳实践**：

xml用来管理bean；

注解只负责完成属性的注入；

**2、使用注解唯一需要注意的就是，必须开启注解的支持**：

​<context:component-scan base-package="com.guo"></context:component-scan>

<context:annotation-config/>

# 二、Spring的常用注解

## 1、给容器中注入组件

**（1）包扫描+组件标注注解**

@Component：泛指各种组件

@Controller、@Service、@Repository都可以称为@Component。

@Controller：控制层

@Service：业务层

@Repository：数据访问层

**（2）@Bean**

导入第三方包里面的注解

**（3）@Import**

@Import(要导入到容器中的组件)；

@ImportSelector：返回需要导入的组件的全类名数组；

@ImportBeanDefinitionRegistrar：手动注册bean到容器中；

**（4）使用spring提供的FactoryBean（工厂Bean）**

* 默认获取到的是工厂Bean调用getObject创建的对象
* 要获取工厂Bean本身，需要在id前面加一个&

## 2、注入bean的注解

@Autowired：由bean提供

@Autowired可以作用在变量、setter方法、构造函数上；

有个属性为required，可以配置为false；

@Inject：由JSR-330提供

@Inject用法和@Autowired一样。

@Resource：由JSR-250提供

@Autowired、@Inject是默认按照类型匹配的，@Resource是按照名称匹配的，@Autowired如果需要按照名称匹配需要和@Qualifier一起使用，@Inject和@Name一起使用。

@Primary

让spring进行自动装配的时候，默认使用首选的bean，和@Qualifier一个效果。

**AutowiredAnnotationBeanPostProcessor用于处理 @Autowired注解**

## 3、@JsonIgnore

（1）作用

在json序列化时将java bean中的一些属性忽略掉，序列化和反序列化都受影响。

（2）使用方法

一般标记在属性或者方法上，返回的json数据即不包含该属性。

（3）注解失效

如果注解失效，可能是因为你使用的是fastJson，尝试使用对应的注解来忽略字段，注解为：@JSONField(serialize = false)，使用方法一样。

## 4、Java配置类相关注解

@Configuration

声明当前类为配置类；

@Bean

注解在方法上，声明当前方法的返回值为一个bean，替代xml中的方式；

@ComponentScan

用于对Component进行扫描；

## 5、切面（AOP）相关注解

Spring AOP详细介绍

Spring支持AspectJ的注解式切面编程。

@Aspect 声明一个切面

@After 在方法执行之后执行（方法上）

@Before 在方法执行之前执行（方法上）

@Around 在方法执行之前与之后执行（方法上）

@PointCut 声明切点

在java配置类中使用@EnableAspectJAutoProxy注解开启Spring对AspectJ代理的支持

## 6、@Bean的属性支持

@Scope设置类型包括：

设置Spring容器如何新建Bean实例（方法上，得有@Bean）

① Singleton

（单例,一个Spring容器中只有一个bean实例，默认模式）,

② Protetype

（每次调用新建一个bean）,

③ Request

（web项目中，给每个http request新建一个bean）,

④ Session

（web项目中，给每个http session新建一个bean）,

⑤ GlobalSession

（给每一个 global http session新建一个Bean实例）

## 7、@Value注解

**（1）支持如下方式的注入：**

注入普通字符

注入操作系统属性

注入表达式结果

注入其它bean属性

注入文件资源

注入网站资源

注入配置文件

**（2）@Value三种情况的用法**:

${}是去找外部配置的参数，将值赋过来

#{}是SpEL表达式，去寻找对应变量的内容

#{}直接写字符串就是将字符串的值注入进去

## 8、环境切换

@Profile

指定组件在哪个环境的情况下才能被注册到容器中，不指定，任何环境下都能注册这个组件。

@Conditional

通过实现Condition接口，并重写matches方法，从而决定该bean是否被实例化。

## 9、异步相关

@EnableAsync

配置类中通过此注解开启对异步任务的支持；

@Async

在实际执行的bean方法使用该注解来声明其是一个异步任务（方法上或类上所有的方法都将异步，需要@EnableAsync开启异步任务）

## 10、定时任务相关

@EnableScheduling

在配置类上使用，开启计划任务的支持（类上）

@Scheduled

来申明这是一个任务，包括cron,fixDelay,fixRate等类型（方法上，需先开启计划任务的支持）

## 11、Enable\*\*\*注解说明

这些注解主要是用来开启对xxx的支持：

@EnableAspectAutoProxy：开启对AspectJ自动代理的支持；

@EnableAsync：开启异步方法的支持；

@EnableScheduling：开启计划任务的支持；

@EnableWebMvc：开启web MVC的配置支持；

@EnableConfigurationProperties：开启对@ConfigurationProperties注解配置Bean的支持；

@EnableJpaRepositories：开启对SpringData JPA Repository的支持；

@EnableTransactionManagement：开启注解式事务的支持；

@EnableCaching：开启注解式的缓存支持；

## 12、测试相关注解

@RunWith

运行器，Spring中通常用于对JUnit的支持

@ContextConfiguration

用来加载配置配置文件，其中classes属性用来加载配置类。

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations = {"classpath\*:/\*.xml"})

public class CDPlayerTest {

}

@ContextConfiguration这个注解通常与@RunWith(SpringJUnit4ClassRunner.class)联合使用用来测试。

@ContextConfiguration括号里的locations = {"classpath\*:/\*.xml"}就表示将classpath路径里所有的xml文件都包括进来，自动扫描的bean就可以拿到，此时就可以在测试类中使用@Autowired注解来获取之前自动扫描包下的所有bean。

## 13、@SuppressWarnings

Suppress 抑制；镇压；废止 Warnings警告

@SuppressWarnings("resource")是J2SE 提供的一个批注。该批注的作用是给编译器一条指令，告诉它对被批注的代码元素内部的某些警告保持静默。

@SuppressWarnings 批注允许您选择性地取消特定代码段（即，类或方法）中的警告。其中的想法是当您看到警告时，您将调查它，如果您确定它不是问题，您就可以添加一个 @SuppressWarnings 批注，以使您不会再看到警告。

虽然它听起来似乎会屏蔽潜在的错误，但实际上它将提高代码安全性，因为它将防止您对警告无动于衷 — 您看到的每一个警告都将值得注意。

# 三、SpringMVC常用注解

1、@EnableWebMv**c**

在配置类中开启Web MVC的配置支持。

2、@Controller

3、@RequestMapping

用于映射web请求，包括访问路径和参数。

4、@ResponseBody

支持将返回值放到response内，而不是一个页面，通常用户返回json数据。

5、@RequestBody

允许request的参数在request体中，而不是在直接连接的地址后面。（放在参数前）

6、@PathVariable

用于接收路径参数，比如@RequestMapping(“/hello/{name}”)声明的路径，将注解放在参数前，即可获取该值，通常作为Restful的接口实现方法。

7、@RestController

该注解为一个组合注解，相当于@Controller和@ResponseBody的组合，注解在类上，意味着，该Controller的所有方法都默认加上了@ResponseBody。

8、@ControllerAdvice

全局异常处理

全局数据绑定

全局数据预处理

ControllerAdvice的常用场景

9、@ExceptionHandler

用于全局处理控制器里的异常。

10、@InitBinder

用来设置WebDataBinder，WebDataBinder用来自动绑定前台请求参数到Model中。

11、@ModelAttribute

（1）@ModelAttribute注释方法

如果把@ModelAttribute放在方法的注解上时，代表的是：该Controller的所有方法在调用前，先执行此@ModelAttribute方法。可以把这个@ModelAttribute特性，应用在BaseController当中，所有的Controller继承BaseController，即可实现在调用Controller时，先执行@ModelAttribute方法。比如权限的验证（也可以使用Interceptor）等。

（2）@ModelAttribute注释一个方法的参数

当作为方法的参数使用，指示的参数应该从模型中检索。如果不存在，它应该首先实例化，然后添加到模型中，一旦出现在模型中，参数字段应该从具有匹配名称的所有请求参数中填充。



hellorWord方法的userLogin参数的值来源于getUserLogin()方法中的model属性。

12、@Transactional

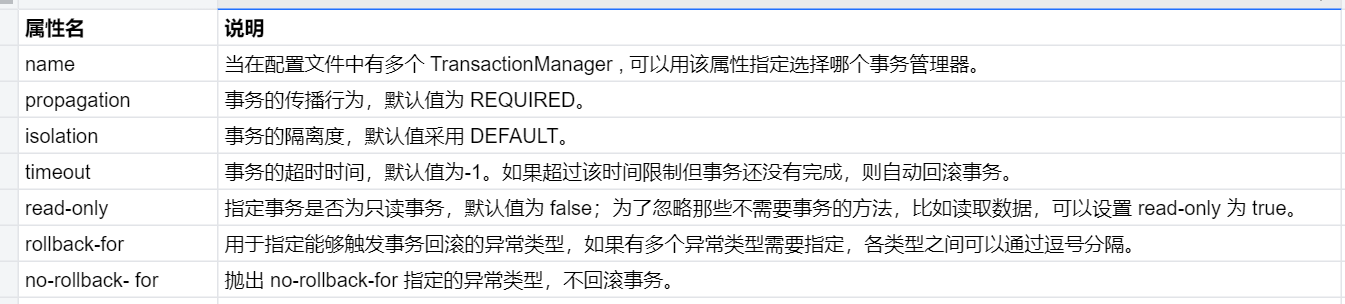
@Transactional 注解放在类级别时，表示所有该类的公共方法都配置相同的事务属性信息。EmployeeService 的所有方法都支持事务并且是只读。当类级别配置了@Transactional，方法级别也配置了@Transactional，应用程序会以方法级别的事务属性信息来管理事务，换言之，方法级别的事务属性信息会覆盖类级别的相关配置信息。

@Transactional(propagation= Propagation.SUPPORTS,readOnly=true)

@Service(value ="employeeService")

public class EmployeeService

@Transactional 注解的属性信息



**TransactionalAnnotationProcessor**用于处理 **@Transactional**注解

# 四、其它注解

1、@JsonIgnore

（1）作用

在json序列化时将java bean中的一些属性忽略掉，序列化和反序列化都受影响。

（2）使用方法

一般标记在属性或者方法上，返回的json数据即不包含该属性。

（3）注解失效

如果注解失效，可能是因为你使用的是fastJson，尝试使用对应的注解来忽略字段，注解为：@JSONField(serialize = false)，使用方法一样。

2、@Schema

@schema注解标注在class上，表示此类对应的数据库表对应的schema。

可以用如下语句判断某个实体类上是否带有@schema注解，从而得到schema。

bean.getClass().isAnnotationPresent(Schema.class)

# 五、json常用注解

**1、@JsonIgnoreProperties**

此注解是类注解，作用是json序列化时将java bean中的一些属性忽略掉，序列化和反序列化都受影响。

写法将此标签加在user类的类名上 ，可以多个属性也可以单个属性。

//生成json时将name和age属性过滤

@JsonIgnoreProperties({"name"},{"age"})

public class user {

private String name;

private int age;

}

**2、@JsonIgnore**

此注解用于属性或者方法上（最好是属性上），作用和上面的@JsonIgnoreProperties一样。

//生成json 时不生成age 属性

public class user {

private String name;

@JsonIgnore

private int age;

}

**3、@JsonFormat**

此注解用于属性或者方法上（最好是属性上），可以方便的把Date类型直接转化为我们想要的模式，比如：

public class User{

@JsonFormat(pattern = “yyyy-MM-dd HH-mm-ss”)

private Date date;

}

4、@JsonSerialize

此注解用于属性或者getter方法上，用于在序列化时嵌入我们自定义的代码，比如序列化一个double时在其后面限制两位小数点。

5、@JsonDeserialize

此注解用于属性或者setter方法上，用于在反序列化时可以嵌入我们自定义的代码，类似于上面的@JsonSerialize

6、@Transient

如果一个属性并非数据库表的字段映射，就务必将其标示为@Transient，否则ORM框架默认其注解为@Basic；

7、@JsonIgnoreType

标注在类上，当其他类有该类作为属性时，该属性将被忽略。

8、@JsonProperty

@JsonProperty 可以指定某个属性和json映射的名称。例如我们有个json字符串为{“user\_name”:”aaa”}，

而java中命名要遵循驼峰规则，则为userName，这时通过@JsonProperty 注解来指定两者的映射规则即可。这个注解也比较常用。

public class SomeEntity {

@JsonProperty("user\_name")

private String userName;

}

9、只在序列化情况下生效的注解

@JsonPropertyOrder

在将 java pojo 对象序列化成为 json 字符串时，使用 @JsonPropertyOrder 可以指定属性在 json 字符串中的顺序。

@JsonInclude

在将 java pojo 对象序列化成为 json 字符串时，使用 @JsonInclude 注解可以控制在哪些情况下才将被注解的属性转换成 json，例如只有属性不为 null 时。

@JsonInclude(JsonInclude.Include.NON\_NULL)

这个注解放在类头上，返给前端的json里就没有null类型的字段，即实体类与json互转的时候 属性值为null的不参与序列化。

另外还有很多其它的范围，例如 NON\_EMPTY、NON\_DEFAULT等

10、在反序列化情况下生效的注解

@JsonSetter

@JsonSetter 标注于 setter 方法上，类似 @JsonProperty ，也可以解决 json 键名称和 java pojo 字段名称不匹配的问题。