Spring2.5 注解介绍（3.0 通用）

Auther: 韩群峰 Version: 1.0.0 Date: 2011-03-15

注解说明

- 注册注解处理器
  - 方式一:bean

```
<bean class="org.springframework.beans.factory.annotation.
AutowiredAnnotationBeanPostProcessor"/>
```

  - 方式二: 命名空间<context:annotation-config />

<context:annotationconfig /> 将隐式地向 Spring 容器注册 AutowiredAnnotationBeanPostProcessor、CommonAnnotationBeanPostProcessor 、 PersistenceAnnotationBeanPostProcessor 以及 RequiredAnnotationBeanPostProcessor 这 4 个 BeanPostProcessor 。

  - 方式三: 命名空间<context:component-scan />

如果要使注解工作，则必须配置 component-scan ，实际上不需要再配置 annotation-config。
base-package 属性指定了需要扫描的类包，类包及其递归子包中所有的类都会被处理。还允许定义过滤器将基包下的某些类纳入或排除。

- Spring 支持以下 4 种类型的过滤方式：
  - 注解 org.example.SomeAnnotation 将所有使用 SomeAnnotation 注解的类过滤出来
  - 类名指定 org.example.SomeClass 过滤指定的类
  - 正则表达式 com.kedacom.spring.annotation.web..* 通过正则表达式过滤一些类
  - AspectJ 表达式 org.example..*Service+ 通过 AspectJ 表达式过滤一些类

- 正则表达式的过滤方式举例：

```
<context:component-scanbase-package="com.casheen.spring.annotation">
<context:exclude-filtertype="reg
ex"
expression="com.casheen.sprin
g.annotation.web..*"/>
</context:component-scan>
```

- 注解的过滤方式举例：

```
<context:component-scan base-package="com.netqin" >
    <context:include-filter type="annotation"
    expression="org.springframework.stereotype.Controller"/>
    <context:include-filter type="annotation"
    expression="org.springframework.stereotype.Service"/>
    <context:include-filter type="annotation"
    expression="org.springframework.stereotype.Repository"/>
</context:component-scan>
```

启用 Spring MVC 注解

• 启动 Spring MVC 的注解功能，完成请求和注解 POJO 的映射

• <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"/>

注解介绍

- @Controller
- @Service
- @Autowired
- @RequestMapping
- @RequestParam
- @ModelAttribute
- @Cacheable
- @CacheFlush
- @Resource
- @PostConstruct
- @PreDestroy
- @Repository
- @Component （不推荐使用）
- @Scope
- @SessionAttributes
- @InitBinder
- @Required
- @Qualifier

@Controller

- 例如

@Controller
public class SoftCreateController extends SimpleBaseController {}

- 或者

@Controller("softCreateController")

- 说明

@Controller 负责注册一个 bean 到 spring 上下文，bean 的 ID 默认为类名称开头字母小写

@Service

- 例如

@Service
public class SoftCreateServiceImpl implements ISoftCreateService {}

- 或者

@Service("softCreateServiceImpl")

- 说明

@Service 负责注册一个 bean 到 spring 上下文，bean 的 ID 默认为类名称开头字母小写

@Autowired

- 例如

@Autowired
private ISoftPMService softPMService;

- 或者

```
@Autowired(required=false)
private ISoftPMService softPMService = new SoftPMServiceImpl();
```

• 说明

@Autowired 根据 bean 类型从 spring 上线文中进行查找，注册类型必须唯一，否则报异常。与@Resource 的区别在于，@Resource 允许通过 bean 名称或 bean 类型两种方式进行查找@Autowired(required=false) 表示，如果 spring 上下文中没有找到该类型的 bean 时， 才会使用 new SoftPMServiceImpl();

@Autowired 标注作用于 Map 类型时，如果 Map 的 key 为 String 类型，则 Spring 会将容器中所有类型符合 Map 的 value 对应的类型的 Bean 增加进来，用 Bean 的 id 或 name 作为 Map 的 key。

@Autowired 还有一个作用就是，如果将其标注在 BeanFactory 类型、ApplicationContext 类型、ResourceLoader 类型、ApplicationEventPublisher 类型、MessageSource 类型上，那么 Spring 会自动注入这些实现类的实例，不需要额外的操作。

@RequestMapping

• 类

```
@Controller
@RequestMapping("/bbtForum.do")
public class BbtForumController {
        @RequestMapping(params = "method=listBoardTopic")
         public String listBoardTopic(int topicId,User user) {}
    }
```

• 方法

```
        @RequestMapping("/softpg/downSoftPg.do")
            @RequestMapping(value="/softpg/ajaxLoadSoftId.do",method = POST)
         @RequestMapping(value = "/osu/product/detail.do", params = { "modify=false" },
method =POST)
```

• 说明

@RequestMapping 可以声明到类或方法上

• 参数绑定说明

如果我们使用以下的 URL 请求：

http://localhost/bbtForum.do?method=listBoardTopic&topicId=1&userId=10&userName=tom
topicId URL 参数将绑定到 topicId 入参上，而 userId 和 userName URL 参数将绑定到 user 对象的 userId 和 userName 属性中。和 URL 请求中不允许没有 topicId 参数不同，虽然 User 的 userId 属性的类型是基本数据类型，但如果 URL 中不存在 userId 参数，Spring 也不会报错，此时 user.userId 值为 0 。如果 User 对象拥有一个 dept.deptId 的级联属性，那么它将和 dept.deptId URL 参数绑定。

@RequestParam

• 参数绑定说明

@RequestParam("id")

http://localhost/bbtForum.do?method=listBoardTopic&id=1&userId=10&userName=tom
listBoardTopic(@RequestParam("id")int topicId,User user) 中的 topicId 绑定到 id 这个 URL 参数， 那么可以通过对入参使用 @RequestParam 注解来达到目的

@RequestParam(required=false)：参数不是必须的，默认为 true
@RequestParam(value="id",required=false)

请求处理方法入参的可选类型

• Java 基本数据类型和 String

默认情况下将按名称匹配的方式绑定到 URL 参数上，可以通过 @RequestParam 注解改变默认的绑定规则

• request/response/session

既可以是 Servlet API 的也可以是 Portlet API 对应的对象，Spring 会将它们绑定到 Servlet 和 Portlet 容器的相应对象上

• org.springframework.web.context.request.WebRequest

内部包含了 request 对象

• java.util.Locale

绑定到 request 对应的 Locale 对象上

• java.io.InputStream/java.io.Reader

可以借此访问 request 的内容

• java.io.OutputStream / java.io.Writer

可以借此操作 response 的内容

• 任何标注了 @RequestParam 注解的入参

被标注 @RequestParam 注解的入参将绑定到特定的 request 参数上。

• java.util.Map / org.springframework.ui.ModelMap

它绑定 Spring MVC 框架中每个请求所创建的潜在的模型对象，它们可以被 Web 视图对象访问（如 JSP ）

• 命令/ 表单对象（注：一般称绑定使用 HTTP GET 发送的 URL 参数的对象为命令对象，而称绑定使用 HTTP POST 发送的 URL 参数的对象为表单对象）

它们的属性将以名称匹配的规则绑定到 URL 参数上，同时完成类型的转换。

而类型转换的规则可以通过 @InitBinder 注解或通过 HandlerAdapter 的配置进行调 整

• org.springframework.validation.Errors / org.springframework.validation.BindingResult

为属性列表中的命令/ 表单对象的校验结果，注意检验结果参数必须紧跟在命令/ 表单对象的后面

• org.springframework.web.bind.support.SessionStatus

可以通过该类型 status 对象显式结束表单的处理，这相当于触发 session 清除其中的通过 @SessionAttributes 定义的属性

请求处理方法返回值的可选类型

• void

此时逻辑视图名由请求处理方法对应的 URL 确定，如以下的方法：

```
@RequestMapping("/welcome.do")
public void welcomeHandler() {}
```

对应的逻辑视图名为 “ welcome ”

• String

此时逻辑视图名为返回的字符，如以下的方法：

```
@RequestMapping(method = RequestMethod.GET)
```

```
public String setupForm(@RequestParam("ownerId") int ownerId, ModelMap model)
{
    Owner owner = this.clinic.loadOwner(ownerId);
    model.addAttribute(owner);
    return "ownerForm";
}
```

对应的逻辑视图名为 “ ownerForm ”

• org.springframework.ui.ModelMap

和返回类型为 void 一样，逻辑视图名取决于对应请求的 URL ，如下面的例子：

```
@RequestMapping("/vets.do")
public ModelMap vetsHandler() {
    return new ModelMap(this.clinic.getVets());
    }
```

对应的逻辑视图名为 “ vets ” ，返回的 ModelMap 将被作为请求对应的模型对象，可以在 JSP 视图页面中访问到。

• ModelAndView

当然还可以是传统的 ModelAndView 。

@ModelAttribute

• 作用域：request

• 例如

```
@RequestMapping("/base/userManageCooper/init.do")
public String handleInit(@ModelAttribute("queryBean") ManagedUser
sUser,Model model,){
```

• 或者

```
@ModelAttribute("coopMap")// 将 coopMap 返回到页 面
public Map<Long,CooperatorInfo> coopMapItems(){}
```

• 说明

@ModelAttribute 声明在属性上，表示该属性的 value 来源于 model 里"queryBean" ，并被保存到 model 里@ModelAttribute 声明在方法上，表示该方法的返回值被保存到 model 里

@Cacheable 和@CacheFlush

- @Cacheable ： 声明一个方法的返回值应该被缓 存

例如：@Cacheable(modelId = "testCaching")

- @CacheFlush ： 声明一个方法是清空缓存的触发器

例如：@CacheFlush(modelId = "testCaching")

- 说明

要配合缓存处理器使用，参考： http://hanqunfeng.iteye.com/blog/603719

spring3.0 没有对缓存提供支持，不过 3.1 之后就有了，可以参考：Spring3.1 Cache 注解

@Resource

- 例如

```
@Resource
    private DataSource dataSource; // inject the bean named 'dataSource'
```

- 或者

```
@Resource(name="dataSource")
@Resource(type=DataSource.class)
```

- 说明

@Resource 默认按 bean 的 name 进行查找，如果没有找到会按 type 进行查找，此时与@Autowired 类 似

在没有为 @Resource 注解显式指定 name 属性的前提下，如果将其标注在 BeanFactory 类型、ApplicationContext 类型、ResourceLoader 类型、ApplicationEventPublisher 类型、MessageSource 类型上，那么 Spring 会自动注入这些实现类的实例，不需要额外的操作。此时 name 属性不需要指定（或者指定为 ""），否则注入失败；

@PostConstruct 和@PreDestroy

- @PostConstruct

在方法上加上注解@PostConstruct，这个方法就会在 Bean 初始化之后被 Spring 容器执 行

（注：Bean 初始化包括，实例化 Bean ，并装配 Bean 的属性（依赖注入））。

• @PreDestroy

在方法上加上注解@PreDestroy ， 这个方法就会在 Bean 被销毁前被 Spring 容器执行。

@Repository

• 与@Controller 、@Service 类似，都是向 spring 上下文中注册 bean ，不在赘述。

@Component（不推荐使用）

• @Component

@Component 是所有受 Spring 管理组件的通用形式，Spring 还提供了更加细化的注解形式： @Repository 、@Service、@Controller ，它们分别对应存储层 Bean ，业务层 Bean ，和展示层 Bean 。

目前版本（2.5 ）中，这些注解与@Component 的语义是一样的，完全通用， 在 Spring 以后的版本中可能会给它们追加更多的语义。所以，我们推荐使用@Repository 、@Service 、@Controller 来替代@Component 。

@Scope

• 例如

```
@Scope("session")
@Repository()
public class UserSessionBean implementsSerializable {}
```

• 说明

在使用 XML 定义 Bean 时，可以通过 bean 的 scope 属性来定义一个 Bean 的作用范围，

同样可以通过@Scope 注解来完成

@Scope 中可以指定如下值：

singleton:定义 bean 的范围为每个 spring 容器一个实例（默认值）

prototype:定义 bean 可以被多次实例化（使用一次就创建一次）

request:定义 bean 的范围是 http 请求（springMVC 中有效）

session:定义 bean 的范围是 http 会话（springMVC 中有效）

global-session:定义 bean 的范围是全局 http 会话（portlet 中有效）

@SessionAttributes

- 说明

Spring 允许我们有选择地指定 ModelMap 中的哪些属性需要转存到 session 中，
以便下一个请求属对应的 ModelMap 的属性列表中还能访问到这些属性。
这一功能是通过类定义处标注 @SessionAttributes 注解来实现的。
@SessionAttributes 只能声明在类上，而不能声明在方法上。

- 例如

```
@SessionAttributes("currUser") // 将 ModelMap 中属性名为 currUser 的属性
@SessionAttributes({"attr1","attr2"})
@SessionAttributes(types = User.class)
@SessionAttributes(types = {User.class,Dept.class})
@SessionAttributes(types = {User.class,Dept.class},value={"attr1","attr2"})
```

@InitBinder

- 说明

如果希望某个属性编辑器仅作用于特定的 Controller，
可以在 Controller 中定义一个标注 @InitBinder 注解的方法，
可以在该方法中向 Controller 了注册若干个属性编辑器

- 例如

```
@InitBinder
public void initBinder(WebDataBinder binder) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    dateFormat.setLenient(false);
    binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat,
    false));

}
```

@Required

• 例如

```
@required
public   setName(String name){}
```

• 说明

@ required 负责检查一个 bean 在初始化时其声明的 set 方法是否被执行，当某个被标注了 @Required 的 Setter 方法没有被调用，则 Spring 在解析的时候会抛出异常，以提醒开发者对相应属性进行设置。 @Required 注解只能标注在 Setter 方法之上。因为依赖注入的本质是检查 Setter 方法是否被调用了，而不是真的去检查属性是否赋值了以及赋了什么样的值。如果将该注解标注在非 setXxxx() 类型的方法则被忽略。

@Qualifier

• 例如

```
@Autowired
 @Qualifier("softService")
 private ISoftPMService softPMService;
```

• 说明

使用@Autowired 时，如果找到多个同一类型的 bean，则会抛异常，此时可以使用 @Qualifier("beanName")，明确指定 bean 的名称进行注入，此时与 @Resource 指定 name 属性作用相同。