# 1. 基于注解的配置

要使注解可用，您必须使用 Java 5 (Tiger)或更新的版本，以使得可以访问源代码层次的注解。这些注解可以被注册为独立 bean 的定义，但它们也可以被隐式地注册，通过基于 XML 的配置方式，如下例（请注意包含 'context' 命名空间）：

注意：注解注入会在XML注入之前执行，因此通过两种方式，那么XML配置会覆盖注解的配置。

<?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:annotation-config/>*

</beans>

（隐式注册 post-processors 包括了 AutowiredAnnotationBeanPostProcessor，CommonAnnotationBeanPostProcessor，PersistenceAnnotationBeanPostProcessor及 RequiredAnnotationBeanPostProcessor。）

注意：<context:annotation-config/>仅仅查找定义在同一个Application Context的Bean的注解。这就意味着，如果你为DispatcerServlet将<context:annotation-config/>放置在WebApplictionContext中，那么它仅仅检查控制器中的@Autowired bean，其它的bean（如服务层bean）不会检查。

## 1.1 Spring MVC注解

### @Controller注解

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

### @Autowired注解

* 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注解

### @RequestParam注解

### @ModelAttribute注解

1）绑定请求参数到命令对象：放在功能处理方法的入参上时，用于将多个请求参数绑定到一个命令对象，从而简化绑定流程，而且自动暴露为模型数据用于视图页面展示时使用。其实@ModelAttribute此处对于供视图页面展示来说与model.addAttribute("attributeName", abc);功能类似。

2） 暴露表单引用对象为模型数据：放在处理器的一般方法（非功能处理方法）上时，是为表单准备要展示的表单引用对象，如注册时需要选择的所在城市等，而且在执行功能处理方法（@RequestMapping 注解的方法）之前，自动添加到模型对象中，用于视图页面展示时使用

3） 暴露@RequestMapping 方法返回值为模型数据：放在功能处理方法的返回值上时，是暴露功能处理方法的返回值为模型数据，用于视图页面展示时使用。

### @RequestMapping注解

### @RequestMapping注解

### @RequestMapping注解

### @RequestMapping注解

## 1.2 Spring Context注解

### @Required

@Required注解应用于bean属性的setter方法中，表明受影响Bean的属性必须在Bean的定义中或者自动装配中填充。如果受影响的Bean属性没有被填充，容器会抛出异常。

**public** **class** SimpleMovieLister {

**private** MovieFinder movieFinder;

*@Autowired*

**public** **void** setMovieFinder(MovieFinder movieFinder) {

**this**.movieFinder = movieFinder;

}

*// ...*

}

### @Component注解

@Component是所有受Spring管理组件的通用形式； 而@Repository、@Service和 @Controller则是@Component的细化， 用来表示更具体的用例：

@Repositor：持久化层，用它来标记充当储存库(又称 Data Access Object或DAO)

@Service：服务层

@Controller：表现层

也就是说，用@Componen来注解你的组件类，但如果用@Repository、@Service 或@Controller来注解它们，你的类也许能更好地被工具处理，或与切面进行关联。 例如，这些典型化注解可以成为理想的切入点目标。当然，在Spring Framework以后的版本中， @Repository、@Service和 @Controller也许还能携带更多语义。如此一来，如果你正在考虑服务层中是该用 @Component还是@Service， 那@Service显然是更好的选择。同样的，就像前面说的那样， @Repository已经能在持久化层中进行异常转换时被作为标记使用了。

### @Service注解

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