Spring 自动装配

自动装配有四种方式：

1. Byname

将与Beanr 的属性相同名字（ID）的其他Bean自动装配到Bean的对应属性，如果没有，不进行装配

1. byType

通过类型检测，将与Bean的属性类型相同的其他Bean装配到Bean的属性，如果没有，不进行装配。

1. constructor;

将与Bean的构造器形参类型相同的其他Bean自动装配到Bean构造器的对应形参。

1. autodetect

先尝试 constructor ，如果失败，尝试 byType

自动装配时可以通过设置Bean的 autowire属性 autowire=”byName”进行声明使用byName方式进行装配。

通过设置 Beans 的default-autowire=”” 可以设置默认的装配方式， bean的autowire属性可以覆盖默认的自动装配方法。

注解装配：其效果与自动装配一样。

Spring 默认禁止注解装配，通过<context:annotation-config /> 启用注解装配。

@Autowired 注解可以对 Bean属性，setter方法，其他方法，构造器方法中所需的形参进行自动装配，装配方式是byType.

Autowired标注的属性或方法必须是可以进行装配的。

@Autowired(required=false)其中required属性可以对属性是否进行装配，进行设置，required=false说明属性可以不被装配，对应属性设为null。Required属性可以对注解的所有属性起作用，当使用autowired注解多个构造器时，只有一个autowired的required属性为true，其他为false，构造器注入时会选择形参最多的构造器进行装配。

@Qualifier注解可以缩小候选Bean的可选范围，可以将 @Autowired注解的属性设置为byName自动装配。

Spring提供的注解是@Autowired，而java标准提供了@Inject注解，可以完全替代@Autowired注解。

@Inject与@Autowired一样，可以自动装配属性，方法，构造器；但是@Inject没有required属性，所以必须存在可以装配的Bean。Provider可以提供延迟注入或注入多个Bean。我们可以通过@Named（” ”）注解对Bean的可选范围进行限定，它和Spring中的@Qualifier的作用十分相似。区别在于@Named通过Bean的ID标识可选的Bean。

自动检测Bean

<context:annotation-scan base-package=“包名”>可以替换<context:annotation-config>,同时允许Spring进行自动检测Bean。通过配置 base-package属性指定自动检测的包，及其子包，<context:annotation-scan>元素会扫描包及子包，并能查找出可以自动注册Spring Bean的类。

默认情况下，可以被自动检测的类被构造注解所标注，其注解如下：

@Component——通用的构造型注解，标识类为Spring组件

@Controller——标识类为Spring MVC controller

@Repository——标识将类定义为数据仓库

@Service——标识将类定义为服务

@使用@Component标注的任意自定义注解

对于使用注解的类，创建实例时，默认将Bean的ID命名为类名，类名的第一个字母变为小写。不过可以在注解时显示的指定Bean的ID，如 @Component（”Bean ID”）

通过<context:include-filter> <context:exclude-fileter>可以对扫描行为进行配置

Spring MVC

前端控制器：DispatcherServlet

Web中的所有请求都会发送给DispatcherServlet，再由它进行请求的分发与处理。

(HandlerMapping)处理器映射：知道将请求发送给哪一个处理器。（通过查看@RequestMapping注解）

控制器(Controller)的工作是：接收请求，进行处理，返回Model和View的名称(url)给DispatcherServlet. 控制器给出的视图名是一个逻辑名，可以通过逻辑名查找真正的视图。

视图解析器(ViewResolver)：可以将逻辑视图名匹配为一个特定的视图实现

在Spring MVC项目中web.xml文件中要进行DispatcherServlet的配置，如下：

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<init-param>定义了DispatcherServlet所使用的Spring上下文，即相应的xml配置文件。如果此属性没有配置，则DispatcherServlet会应用位于 WEB-INF下的

(servlet name)dispatcher-servlet.xml应用上下文。

<url-pattern> /说明了DispatcherServlet可以处理所有的请求，包括静态资源。

<mvc:resources mapping=“ ” location=“ ”>可以建立了一个服务于静态资源的处理器，location表明文件所在位置，mapping表明映射到的url路径。

DispatcherServlet的应用上下文配置：（基于注解驱动的Spring MVC开发）

1. 首先进行处理器映射的配置，说明了DispatcherServlet查询的是哪一个处理器映射，Spring自带的处理器映射如下：
2. BeanNameUrlHandlerMapping：根据控制器Bean的名字将控制器映射到URL。
3. ControllerBeanNameHandlerMapping：与BeanNameUrlHandlerMapping类似，根据控制器Bean的名字将控制器映射到URL。使用该处理器映射实现，Bean的名字不需要遵循URL的约定。
4. ControllerClassNameHandlerMapping：通过使用控制器的类名作为URL基础将控制器映射到URL。
5. DefaultAnnotationHandlerMapping：将请求映射给使用@Request-Mapping注解的控制器和控制器方法。
6. SimpleUrlHandlerMapping：使用定义在Spring应用上下文的属性集合将控制器映射到URL。

如果应用上下文中没有找到处理器映射，则DispatcherServlet会创建并使用BeanNameUrlHandlerMapping和DefaultAnnotationHandlerMapping。

1. 进行注解驱动的相关配置

<mvc:annotation-driven /> 通过此标签，可以得到spring MVC所提供的注解驱动特性。

1. 进行控制器的编写。
2. 使用@Controller标注需要进行自动装配的控制器。
3. 通过@RequestMapping({“url1”,”url2”,...}) 标注类中方法是一个请求处理方法，同时它指明了这个方法要处理 “url1”或者“url2”路径所发来的请求。
4. 请求方法接收参数，进行处理，返回视图逻辑名。
5. @RequestMapping(“url”)注解可以标注类，在类级别上对类进行标注说明此控制器可以处理/url路径下的所有请。如果对类中的方法进行@RequestMapping(“url2”)的注解，则此方法可以处理 /url/url2路径下发来的请求。方法级别上的RequestMapping注解缩小了请求的范围。
6. @RequestMapping(value=”url”,method=GET/POST)注解可以表明方法处理哪一个请求.
7. @RequestMapping(method=GET, params=”new”)标注请求处理方法时，则请求处理方法会处理类处理的请求包含名为 new的查询参数。（个人认为查询参数指的就是 request.getParamater(“new”)中页面中名为new 的控件所对应的值）. Params属性能够限制请求处理方法只对包含特定参数的请求进行处理。

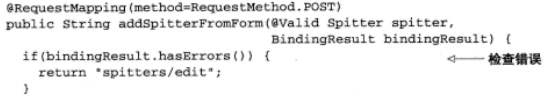
http://localhost:8080/Spitter/spitters?new

@RequestParam(“id”) 注解可以将请求参数通过注解赋值给请求处理方法的参数。

Model对象是在视图与处理器之间传递参数的模型数据，一般为Map<String, Object>类型。可以通过 model.addAttribute(Object obj)方法填充模型。相当于Map中的put方法，不过它的key值是通过计算得到，命名规则是javaBean的命名规则。

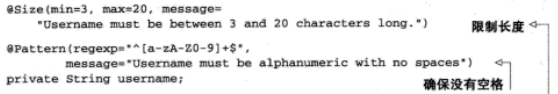
return “redirect:/spitters/”; 请求处理方法在返回的字符中加入 redirect 表明请求被重定向到指定的页面。通过重定向可以避免表单的重复提交

@Valid 注解可以检查参数的合法性



如图：如果spitter对象出错，检验错误会作为第二个参数以BindingResult的形式传递给处理请求的方法，通过对它的hasErrors()方法进行判断可以知道参数是否有错（true---->有错）。

定义检验规则：利用注解可以对数据进行检验规则的定制



1. 解析视图

配置视图解析器，Spring MVC自带的视图解析器如下：

1. BeanNameViewResolver
2. ContentNegotiatingViewResolver
3. FreeMarketViewResolver
4. InternalResourcesViewResolver

在Web应用程序的WAR文件中查找视图模板，视图模板的路径根据加完前缀和后缀的逻辑视图名来确定。

1. JasperReportsViewResolver
2. ResourcesBundleViewResolver
3. TilesViewResolver
4. UrlBasedViewResolver
5. VelocityLayoutViewResolver
6. VelocityViewResolver
7. XsltViewResolver

<!-- 定义ViewResolver 拼凑完整的URL -->

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/views/" />

<property name="suffix" value=".jsp" />

</bean>

Prefix:通过这个前缀指明视图在应用程序所在路径。

注意：在web项目中，虚拟根路径 / 表示 WebContent/

1. 配置DispatcherServlet使用其他的xml配置文件，（一般将配置文件分为服务层，持久层，数据源配置）。

ContextLoaderListener是一个Servlet监听器，可以加载其配置文件到一个Spring上下文中。在web.xml文件中配置listener，如下：

<listener>

<listenerclass>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

在servlet上下文中配置contextConfigLocation参数

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/spitter-servlet.xml

Classpath:service-context.xml

Classpath:persistence-context.xml

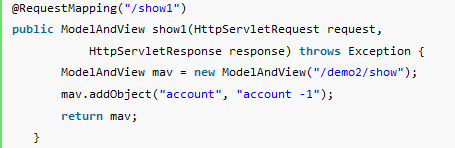
Classpath:dataSource-context.xml

</param-value>

<context-param>

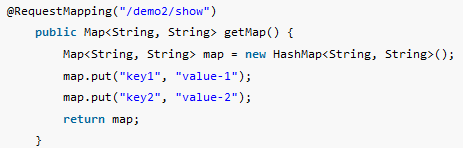
Spring mvc :参数传递 Model ModelAndView Map View String Void

1. ModelAndView



通过ModelAndView构造方法可以指定返回的页面名称，也可以通过setViewName()方法跳转到指定的页面，使用addObject()设置需要返回的值，addObject()有几个不同的参数的方法，可以默认和指定返回对象的名字。调用addObject() 应法将值设置到一个名为ModelMap的类属性，ModelMap是LinkedHashMap的子类。

1. Model 是一个接口，其实现类是一个ExtendedModeMap，继承了ModelMap类。
2. Map 在jsp页面中可以通过 ${key2} 获得值，map.put() 相当于request.addAttribute()方法。



1. String 指定返回的逻辑视图页面名称

