Spring MVC

# 常用注解

@**Controller** **注解在类上面**，表明这个类是SpringMVC里面的Controller，将其声明为Spring的一个bean，Dispatcher servlet 会自动扫描注解了此注解的类，并将Web请求映射到注解了@RequestMapping的方法

@**RequestMapping** 注解是用来映射Web请求（访问路径和参数）、处理类和方法。此注解可以放**在类或者方法**上。注解在方法上的@RequestMapping路径会继承注解在类上得路径。@RequestMapping 支持servlet的request和response作为参数，也支持对request和response的媒体类型进行配置

@**RequestBody** 允许请求的参数在request 中，而不是直接连接在地址后面，此**注解放置在参数前**。

@**PathVariable** 用来接收路径参数，此注**解放置在参数前**

@**ResponseBody** 支持将返回值放置在response体内，而不是返回一个页面，此注解可放置在返**回值或者方法**上。

@**RestController** 相当于@Controller和@ResponseBody

# 基本配置

SpringMVC的定制配置需要继承**WebMvcConfigurerAdapter**类，并在添加@EnableWebMvc注解来开启对对SpringMVC的配置支持，重写该类中的方法可以完成常用配置。

1. **静态资源映射**

静态文件(js,css,图片)等需要直接访问，可以在配置里面重写addResourceHandlers方法实现

addResourceHandler指的是对外暴露的访问路径

addResourceLocations是指文件放置的目录

2. **拦截器配置**

通过重写addInterceptors方法来注册自定义的拦截器。

自定义的拦截器续集成HandleInterceptorAdapter类，其中的

preHandle方法，在请求发生前执行

postHandle方法，在请求完成后执行

3.**@ControllerAdvice**

通过注解了@ControllerAdvice可以将控制器的全局配置放在同一个位置，可以使用以下注解到方法上

**@ExceptionHandler** : 用于全局处理控制器里的异常

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

@ModelAttribute : 用来绑定键值对到Model里

4. ViewController

重写addViewControllers方法

5.路径参数匹配

重写configurePathMatch方法

路径参数中如果带“.”的话，“.”后面的值将会被忽略，如不想忽略，可通过configurer.setUseSuffixPatternMatch(false)

6. **ViewResolver** 接口

Spring MVC 中的接口ViewResolver，实现这个接口要重写resolveViewName() 方法，这个方法的返回值是接口View，而view的职责就是使用 model,request,response对象，并将渲染的视图返回给浏览器。

# 文件上传配置

在Spring 控制器中，

通过MultipartFile file 来接受单个文件

通过MulitpartFile[] files接受多个文件上传

# HttpMessageConvertor

HttpMessageConvertor 是用来处理request和response里的数据，例如有Spring内置的MappingJackson2HttpMessageConvertor,StringHttpMessageConvertor等。

自定义HttpMessageConvertor:

1.继承**AbstractHttpMessageConvertor**接口

2.新建一个新的自定义的媒体类型application/xxxx

2.重写**readInternal** 方法，处理请求的数据

3.重写**writeInternal** 方法，处理如何输出数据到response

4.重写**supports** 方法，说明要处理的类

5.重写**extendMessageConvertors** 方法去添加一个自定义的HttpMessageConvertor，不会覆盖默认的注册的HttpMessageConvertors

# 服务器端推送技术

服务器端推送方案都是基于：当客户端向服务器发送请求，服务器端会抓住这个请求不放，等有数据更新时才返回给客户端，当客户端接收到消息后，再向服务器端发送请求周而复始。

1.基于SSE（Server Send Event）的服务器端推送

（1）服务器端需志明输出的媒体类型为text/event-stream，这是服务器端SSE支持的。

（2）客户端页面添加SSE客户端监听(通过EventSource对象)，在此获得服务器端推送的消息。(只有新式的浏览器才有EventSource对象)

2.基于Servlet 3.0+异步方法处理

（1）servlet.setAsyncSupported(true) 去开启异步方法支持

（2）异步任务的实现是通过控制器从另外一个线程返回一个DeferredResult，所以要第一点，设置定时任务，定时更新DeferredResult，第二点，返回给客户端DeferredResult

可以通过@Scheduled注解某个方法去定时更新DeferredResult, 使用@Scheduled需要在spring启动类前加@EnableSchedulings

（3）在页面使用jQuery的Ajax请求，完成一次请求后就再向后台端发送请求。

# Spring MVC 测试

1. 添加spring-test依赖，可通过设置scope是test时指明这些包的只存在于test周期

2.定义测试类，并

（1）@WebAppConfiguration注解在类上，声明加载的ApplicationContext是一个WebApplicationContext，它的属性指定的是Web资源的位置。

（2）MockMvc ： 模拟MVC对象，通过MockMvcBuilder.webAppContextSetup(this.wac).build() 初始化

（3）@Before 在初始开始前进行初始化工作

（4）@Test 测试方法，对mockMvc对象进行测试操作。

# Spring MVC 注解类型的优点

(1) 一个控制器可以处理多个动作，允许将相关操作写在同一个控制器内。

1. 基于注解的控制器的请求映射不需要存储在配置文件中，使用RequestMapping注释类型，可以对一个方法进行请求处理。

s

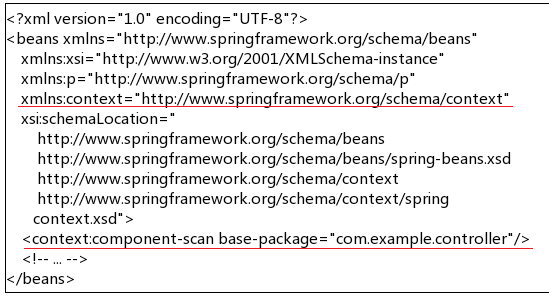
常用类详细解释

**org.springframework.stereotype.Controller**注解类型用于指示Spring类的实例是一个**控制器**。为了保证Spring能找到控制器，就需要

1, 需要在SpringMVC的配置文件中声明spring-context

2, 需要指明<component-scan />元素

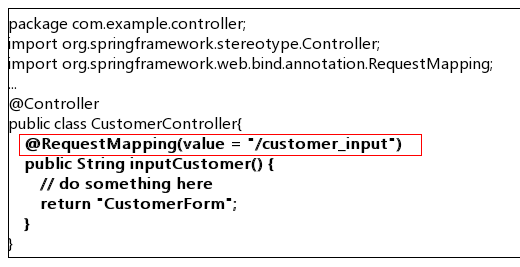
如图



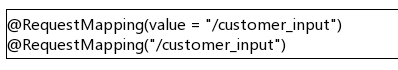
**org.springframework.web.bind.annotation.RequestMapping**注释类型用于映射一个请求和一个方法，可以使用@RequestMapping注释一种方法和类。

一个采用@RequestMapping注释的方法将成为一个请求处理方法，并由调度程序在接收到对应URL请求时调用。RequestMapping注解的value属性值将URL映射到方法。

<http://domain/context>/customer\_input

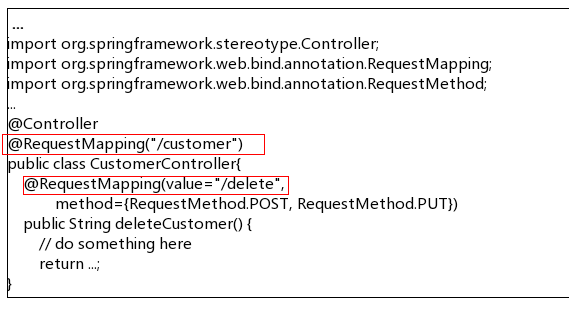


value属性是RequestMapping注释的默认属性，若只有唯一的属性时可以省略属性名称。若超过一个属性时，就必须写入value属性名称。请求映射的value值可以是一个空字符串，此时该方法被映射到 <http://domain/context> 网址。



RequestMapping注释类型也可以用来注释一个控制器(类)。此时，所有的方法都将映射为相对于类级别的请求。

<http://domain/context/customer/delete>



RequestMapping请求处理方法参数支持的**参数类型**：

javax.servlet.ServletRequest

javax.servlet.http.HttpServletRequest

javax.servlet.ServletResponse javax.servlet.http.HttpServletResponse

javax.servlet.http.HttpSession

org.springframework.web.context.request.WebRequest

org.springframework.web.context.request.NativeWebRequest

java.util.Locale

java.io.InputStream

java.io.Reader

java.io.OutputStream

java.io.Writer

java.security.Principal

HttpEntity<?>

java.util.Map

org.springframework.ui.Model

org.springframework.ui.ModelMap

org.springframework.web.servlet.mvc.support.RedirectAttributes

org.springframework.validation.Errors

org.springframework.validation.BindingResult

命令或表单对象

org.springframework.web.bind.support.SessionStatus

org.springframework.web.util.UriComponentsBuilder

带@PathVariable, @MatrixVariable注释的对象

@RequestParam, @RequestHeader, @RequestBody，@RequestPart

请求处理方法支持的返回对象类型：

ModelAndView

Model

Map包含模型的属性

View

代表逻辑视图名的String

Void

提供对Servlet的访问，以响应HTTP头部和内容HttpEntity或ResponseEntity对象

Callable

DeferredResult

其他任意类型，Spring将其视作输出给View的对象模

**@RestController** 与 **@Controller**的区别

@RestController的编写方式依赖注解组合，@RestController被@Controller和@ResponseBody标注，表示@RestController具有两者的注解语义，因此在注解处理时@RestController比@Controller多具有一个@ResponseBody语义，这就是@RestController和@Controller的区别，也是@RestController的返回值为何都是经过转换的json的原因。

所以小结就是：@RestController = @Controller + @ResponseBody