

SpringMVC上传文件时,需要配置MultipartResolver处理器 -->

property name="defaultEncoding" value="UTF-8" />

property name="maxUploadSize" value="15728640" />

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

!-- 指定所上传文件的总大小不能超过15M。注意maxUploadSize属性的限制不是针对单个文件,而是所有文件的容量之和:

<u>ean</u> id="multipartResolver"

1-9都需要从工厂中获得对应类型或者id的bean

1) initMultipartResolver(context);

2) initLocaleResolver(context);默认没

有配置id=localeResolver的bean,则利用

defaultStrategies字段到默认文件中(全 局搜DispatcherServlet.properties)找到

3) initThemeResolver(context);同理没有配置

的bean,也到DispatcherServlet.properties文件

4)initHandlerMappings如果工厂中无HandlerMapping 则也从DispatcherServlet.properties文件中找到默

5)initHandlerAdapters 同initHandlerMappings

8)initViewResolvers 同initHandlerMappings

:interceptors>

<mvc:interceptor>

</mvc:interceptor>

<mvc:interceptor>

正拦截逻辑

c<u>es</u> mapping="/views/**" location="/views/'

中解析该节点,主要围绕注册SimpleUrlHandlerMapping这个bean进行工作

5)注册ResourceHttpRequestHandler,注入属性UrlPathHelper和节点中的

6) 创建SimpleUrlHandlerMapping 加入属性UrlPathHelper、urlMap(key为 节点中的mapping属性、value为第5个beanName)、设置order属性为 Ordered.LOWEST_PRECEDENCE-1、设置属性corsConfigurations

<u>afterPropertiesSet</u>

以上属性在处理请求时会用到,以上三个resources节点会注册3个

resources mapping="/js/**" location="/js/" />

ResourcesBeanDefinitionParser, ResourcesBeanDefinitionParser

resources mapping= $^{''}/$ css/**'' location=''/css/

关于resources的解析MvcNamespaceHandler中注册了

2)注册ResourceUrlProviderExposingInterceptor(一种

并为其添加属性ResourceUrlProviderExposingInterceptor

1)注册ResourceUrlProvider bean

为其加入属性ResourceUrlProvider

4)注入AntPathMatcher、UrlPathHelper bean

location以及contentNegotiationManager属性

3)注册MappedInterceptor bean

SimpleUrlHandlerMapping bean

以上标紫处注册了3中HandlerAdapter

以上标黄处注册了3ExceptionResolver

自的 AbstractHandlerMethodMapping的

会解解析RequestMapping注解

RequestMappingHandlerMapping,又继承了

以上标红处注册了3中HandlerMapping,其中继承

InitializingBean在afterPropertiesSet方法中

HandlerInterceptor)

5)initHandler<mark>A</mark>dapters(context);

8)initViewRes<mark>o</mark>lvers(context); 9)initFlashMapManager(context);

:annotation-driven validator="validator">

<bean class="com.tiangou.tgframe.spring.CustomMappingJacksonConver-</pre>

<value>application/json;charset=UTF-8</value>

关于annotation-driven的解析MvcNamespaceHandler中注册了

corsConfigurations LinkedHashMap 2)注册该类型bean: RequestMappingHandlerAdapter设置

responseBodyAdvice=JsonViewResponseBodyAdvice等属性

DefaultHandlerExceptionResolver(order=2) 异常解析bean

BeanNameUrlHandlerMapping HttpRequestHandlerAdapter.

interceptor=ConversionServiceExposingInterceptor 5)注册ExceptionHandlerExceptionResolve(order=0)、

CompositeUriComponentsContributorFactoryBean

ControllerAdviceBean

AnnotationDrivenBeanDefinitionParser中解析该节点会注册以下bean

contentNegotiationManager=ContentNegotiationManagerFactoryBean

7)最后MvcNamespaceUtils.registerDefaultComponents方法中还会注册

1)注册RequestMappingHandlerMapping这个bean,为这个bean设置属性:

AnnotationDrivenBeanDefinitionParser,

messageConverters=messageConverters

4)注册MappedInterceptor,设置属性:

6) 其它bean如:

以ExceptionHandlerExceptionResolver为例,该类继

承自InitializingBean,其afterPropertiesSet方法中

获取所有带有ControllerAdvice注解的的Bean,并解析其

initExceptionHandlerAdviceCache

金山云常用的异常处理架构

解析带有ControllerAdvice注解的类方法

注解内容,创建ControllerAdviceBean类对象

遍历每个ControllerAdviceBean对象,创建对应

this.exceptionHandlerAdviceCache.put (ControllerAdviceBean, MethodResolver);

设置默认参数解析器、返回值解析

器this.argumentResolvers

进行初始化

MethodResolver

afterPropertiesSet

requestBodyAdvice=sonViewRequestBodyAdvice

ResponseStatusExceptionResolver(order=1),

ResponseStatusExceptionResolver等

SimpleControllerHandlerAdapter

<mvc:message-converters>

</property>

</mvc:message-converters>

c:annotation-driven>

property name="suppot

6) initHandlerExceptionResolvers(context);
7) initRequestToViewNameTranslator(context);

HandlerMappings是如何注册到工厂中的?

ceptionHandlerMethodResolv

解析异常处理bean class中的每个method

this.mappedMethod\$这个map中

解析method中注解指定的exception、方法参数中的

throwable 作为key,当前method作为value存入

天狗用到的有以下两处

9)initFlashMapManager 同initFlashMapManager

6)initHandlerExceptionResolvers 同initHandlerMappings 7)initRequestToViewNameTranslator 同initHandlerMappings

<mvc:mapping path="/redis/**" />

<mvc:mapping path="/privates/**" />

⟨bean class="com.tgou.aop.PublicRedisInterceptor" />

<mvc:exclude-mapping path="/privates/kongdao/tasks/distr:</pre>

<mvc:exclude-mapping path="/privates/kongdao/tasks/clean</pre>

'bean class="com. tiangou. tgframe. aop. LoginHandleI

以上解析由MvcNamespaceHandler,其中注 册了InterceptorsBeanDefinitionParser

InterceptorsBeanDefinitionParser中会将每个interceptor封装为MappedInterceptor注册到工厂中,MappedInterceptor继承自HandlerInterceptor,同时内部又包含HandlerInterceptor(就是我们手写的intercaptor),内部的HandlerInterceptor用于处理真

用于解析interceptors

stractHandlerMethodMapping

遍历所有bean, 查找带有Controller

getMappingForMethod(method, beanType)

---RequestMappingInfo-----

registerHandlerMethod(Object handler,
 Method method, RequestMappingInfo):

或者RequestMapping注解的bean

遍历bean的每<mark>个</mark>method,解析 method上的RequestMapping注解 RequestMappingHandlerMapping

解析并合并method和class的RequestMapping

注解内容,返回RequestMappingInfo

MappingRegistry

1)createHandlerMethod,根据method和beanName创建

HandlerMethod,并解析method参数

2)mappingLookup中注册

key:RequestMappingInfo

Value:RequestMappingInfo

Value:RequestMappingInfo

key:RequestMappingInfo

Value:new MappingRegistration<T>(mapping,

handlerMethod, directUrls, name)

Value:HandlerMethod

3)urlLookup中注册

4)nameLookup中注册 Key:生成的nameString

5) registry中注册

key:url

天狗中有配置,会从此获得

对应的LocaleResolver

中找到对应的LocaleResolver

认的HandlerMapping