## 日志分为：

默认日志：

# 日志文件名，比如：roncoo.log，或者是 /var/log/roncoo.log

logging.file=roncoo.log

# 日志级别配置，比如： logging.level.org.springframework=DEBUG

logging.level.\*=info

logging.level.org.springframework=DEBUG

Lomback:

Log4g2:比log4g好

性能比较：Log4J2 和 Logback 都优于 log4j（不推荐使用）

配置方式：Logback最简洁，spring boot默认，推荐使用



切面日志，打印controller请求和响应参数

## 启动

**java -jar myapp.jar** **--spring.profiles.active=dev**

## 错误处理

**方法一：Spring Boot 将所有的错误默认映射到/error，** **实现ErrorController**

@Controller

@RequestMapping(value = "error")

public class BaseErrorController implements ErrorController {

private static final Logger logger = LoggerFactory.getLogger(BaseErrorController.class);

@Override

public String getErrorPath() {

logger.info("出错啦！进入自定义错误控制器");

return "error/error";

}

@RequestMapping

public String error() {

return getErrorPath();

}

}



**方法二：添加自定义的错误页面**

2.1 html静态页面：在resources/public/error/ 下定义

如添加404页面： resources/public/error/404.html页面，中文注意页面编码

2.2 模板引擎页面：在templates/error/下定义

如添加5xx页面： templates/error/5xx.ftl

注：templates/error/ 这个的优先级比较 resources/public/error/高



**方法三：使用注解@ControllerAdvice**

/\*\*

\* 统一异常处理

\*

\* @param exception

\* exception

\* @return

\*/

@ExceptionHandler({ RuntimeException.class })

@ResponseStatus(HttpStatus.OK)

public ModelAndView processException(RuntimeException exception) {

logger.info("自定义异常处理-RuntimeException");

ModelAndView m = new ModelAndView();

m.addObject("roncooException", exception.getMessage());

m.setViewName("error/500");

return m;

}

/\*\*

\* 统一异常处理

\*

\* @param exception

\* exception

\* @return

\*/

@ExceptionHandler({ Exception.class })

@ResponseStatus(HttpStatus.OK)

public ModelAndView processException(Exception exception) {

logger.info("自定义异常处理-Exception");

ModelAndView m = new ModelAndView();

m.addObject("roncooException", exception.getMessage());

m.setViewName("error/500");

return m;

}



## web 应用开发-Servlets, Filters, listeners

方法一：通过注册 ServletRegistrationBean、 FilterRegistrationBean 和

ServletListenerRegistrationBean 获得控制

方法二：通过实现 ServletContextInitializer 接口直接注册

方法三：在 SpringBootApplication 上使用@ServletComponentScan 注解后，直接通过@WebServlet、@WebFilter、@WebListener 注解自动注册



## CORS 支持



## 文件上传

文件：<input type="file" name="roncooFile" />

<input type="submit" value="上传" />

</form>

@Controller

@RequestMapping(value = "/file")

public class FileController {

private static final Logger logger = LoggerFactory.getLogger(FileController.class);

@RequestMapping(value = "upload")

@ResponseBody

public String upload(@RequestParam("roncooFile") MultipartFile file) {

if (file.isEmpty()) {

return "文件为空";

}

// 获取文件名

String fileName = file.getOriginalFilename();

logger.info("上传的文件名为：" + fileName);

// 获取文件的后缀名

String suffixName = fileName.substring(fileName.lastIndexOf("."));

logger.info("上传的后缀名为：" + suffixName);

// 文件上传路径

String filePath = "d:/roncoo/ttt/";

// 解决中文问题，liunx 下中文路径，图片显示问题

// fileName = UUID.randomUUID() + suffixName;

File dest = new File(filePath + fileName);

// 检测是否存在目录

if (!dest.getParentFile().exists()) {

dest.getParentFile().mkdirs();

}

try {

file.transferTo(dest);

return "上传成功";

} catch (IllegalStateException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return "上传失败";

}

}

三、配置

spring.http.multipart.enabled=true #默认支持文件上传.

spring.http.multipart.file-size-threshold=0 #支持文件写入磁盘.

spring.http.multipart.location= # 上传文件的临时目录

spring.http.multipart.max-file-size=1Mb # 最大支持文件大小

spring.http.multipart.max-request-size=10Mb # 最大支持请求大小