

[spring&Springmvc&Springboo&mybatis]

[Spring&SpringMvc&SpringBoot&Mybatis学习文档]



2018-10-25

[公司名称]

[公司地址]

目录

[1 Spring 2](#_Toc528658465)

[1.1项目启动脚本及顺序（CentOS7、Eclipse） 2](#_Toc528658466)

[1.2项目启动顺序及端口 3](#_Toc528658467)

[1.3虚拟机web 3](#_Toc528658468)

[2 SpringMVC 3](#_Toc528658469)

[2.1为什么要使用Maven？ 3](#_Toc528658470)

[2.2后台工程搭建分析 4](#_Toc528658471)

[3 SpringBoot 4](#_Toc528658472)

[2.1为什么要使用Maven？ 4](#_Toc528658473)

[2.2后台工程搭建分析 4](#_Toc528658474)

[4 Mybatis 4](#_Toc528658475)

[2.1为什么要使用Maven？ 4](#_Toc528658476)

[2.2后台工程搭建分析 4](#_Toc528658477)

[5 SSM 4](#_Toc528658478)

[2.1为什么要使用Maven？ 4](#_Toc528658479)

[2.2后台工程搭建分析 5](#_Toc528658480)

[6 SSM+ 5](#_Toc528658481)

[2.1为什么要使用Maven？ 5](#_Toc528658482)

[2.2后台工程搭建分析 5](#_Toc528658483)

[7 全局RestAPI统一返回 5](#_Toc528658484)

[7.1 BaseResult 5](#_Toc528658485)

[2.2 BasePageResult 8](#_Toc528658486)

[8 全局异常处理 10](#_Toc528658487)

[8.1 @RestControllerAdvice 10](#_Toc528658488)

[8.2异常拦截器 11](#_Toc528658489)

# 1 Spring

## 1.1项目启动脚本及顺序（CentOS7、Eclipse）

**虚拟机：**

**防火墙：**

systemctl status firewalld

systemctl start firewalld

**① systemctl stop firewalld**

**Zookeeper:**

cd /usr/local/zookeeper/zookeeper-3.4.6/bin/

./zkServer.sh stop

**② ./zkServer.sh start**

**Dubbo-Admin:**

cd /opt/apache-tomcat-7.0.59/bin/

**③ ./startup.sh**

**FastDFS:**

**④ /etc/init.d/fdfs\_trackerd start**

**⑤ /etc/init.d/fdfs\_storaged start**

**Nginx:**

**⑥ /usr/local/nginx/sbin/nginx**

**Redis:**

**单机版：**

cd /usr/local/redis/bin/

**⑦ ./redis-server redis.conf**

**集群版：**

cd /usr/local/redis-cluster/

**⑧ ./** **start-all.sh**

**⑨** **./redis-trib.rb create --replicas 1 192.168.188.128:7001 192.168.188.128:7002 192.168.188.128:7003 192.168.188.128:7004 192.168.188.128:7005 192.168.188.128:7006**

**ActiveMQ：**

**🔟** cd /opt/apache-activemq-5.12.0/bin

**./activemq start**

## 1.2项目启动顺序及端口

**项目：**

## 1.3虚拟机web

**Dubbo-admin:**

# 2 SpringMVC

## 2.1为什么要使用Maven？

## 2.2后台工程搭建分析

# 3 SpringBoot

## 2.1为什么要使用Maven？

## 2.2后台工程搭建分析

# 4 Mybatis

## 2.1为什么要使用Maven？

## 2.2后台工程搭建分析

# 5 SSM

## 2.1为什么要使用Maven？

## 2.2后台工程搭建分析

# 6 SSM+

## 2.1为什么要使用Maven？

## 2.2后台工程搭建分析

# 7 全局RestAPI统一返回

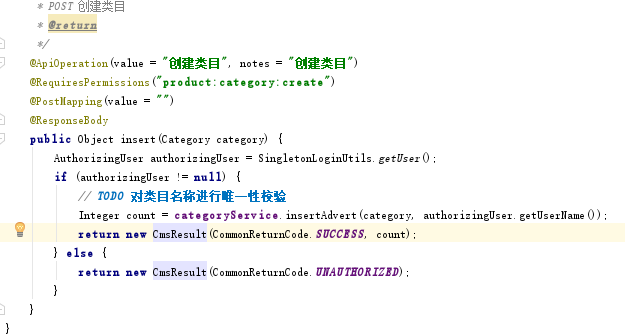
## 7.1 BaseResult

**public class** BaseResult **implements** Serializable{  
   
 **private static final long *serialVersionUID*** = 1L;  
  
 */\*\* 返回状态码 \*/* **private** Integer **code**;  
   
 */\*\* 返回信息 \*/* **private** String **message**;  
   
 */\*\* 返回数据 \*/* **private** Object **data**;  
   
 **public** BaseResult(Integer code, String message) {  
 **super**();  
 **this**.**code** = code;  
 **this**.**message** = message;  
 }  
  
 **public** BaseResult(Integer code, String message, Object data) {  
 **super**();  
 **this**.**code** = code;  
 **this**.**message** = message;  
 **this**.**data** = data;  
 }  
  
 **public** Integer getCode() {  
 **return code**;  
 }  
  
 **public void** setCode(Integer code) {  
 **this**.**code** = code;  
 }  
  
 **public** String getMessage() {  
 **return message**;  
 }  
  
 **public void** setMessage(String message) {  
 **this**.**message** = message;  
 }  
  
 **public** Object getData() {  
 **return data**;  
 }  
  
 **public void** setData(Object data) {  
 **this**.**data** = data;  
 }  
}

**public class** CmsResult **extends** BaseResult {  
  
 **private static final long *serialVersionUID*** = 1L;  
  
 **public** CmsResult(ReturnCode returnCode) {  
 **super**(returnCode.getCode(), returnCode.getMessage());  
 }  
   
 **public** CmsResult(ReturnCode returnCode, Object data) {  
 **super**(returnCode.getCode(), returnCode.getMessage(), data);  
 }  
   
 **public** CmsResult(Integer code, String message) {  
 **super**(code, message);  
 }  
}

**public interface** ReturnCode {  
   
 **public** Integer getCode();  
  
 **public** String getMessage();  
   
}

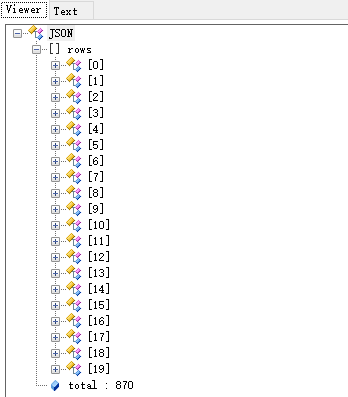
**public enum** CommonReturnCode **implements** ReturnCode {  
   
 */\*\* 请求失败 \*/* ***FAILED***(0, **"failed"**),  
 */\*\* 请求成功 \*/* ***SUCCESS***(1, **"success"**),  
 */\*\* 未知错误 \*/* ***UNKNOWN\_ERROR***(-1, **"未知错误,请联系管理员!"**),  
   
 */\*\* 200请求成功 \*/* ***OK***(200, **"请求成功"**),  
 */\*\* 207频繁操作 \*/* ***MULTI\_STATUS***(207, **"频繁操作"**),  
   
 */\*\* 303登录失败 \*/* ***LOGIN\_FAIL***(303, **"登录失败 "**),  
   
 */\*\* 400请求参数出错 \*/* ***BAD\_REQUEST***(400, **"请求参数出错"**),  
 */\*\* 401没有登录 \*/* ***UNAUTHORIZED***(401, **"您未登录或者登录已超时,请先登录!"**),  
 */\*\* 403没有权限 \*/* ***FORBIDDEN***(403, **"没有权限"**),  
 */\*\* 404找不到页面 \*/* ***NOT\_FOUND***(404, **"找不到页面"**),  
 */\*\* 408请求超时 \*/* ***REQUEST\_TIMEOUT***(408, **"请求超时"**),  
 */\*\* 409发生冲突 \*/* ***CONFLICT***(409, **"发生冲突"**),  
 */\*\* 410已被删除 \*/* ***GONE***(410, **"已被删除"**),  
 */\*\* 423已被锁定 \*/* ***LOCKED***(423, **"已被锁定"**),  
   
 */\*\* 500服务器出错 \*/* ***INTERNAL\_SERVER\_ERROR***(500, **"服务器出错"**);  
   
 */\*\* 返回状态码 \*/* **private** Integer **code**;  
  
 */\*\* 返回消息 \*/* **private** String **message**;  
   
 **private** CommonReturnCode(Integer code, String message) {  
 **this**.**code** = code;  
 **this**.**message** = message;  
 }  
  
 @Override  
 **public** Integer getCode() {  
 **return code**;  
 }  
  
 @Override  
 **public** String getMessage() {  
 **return message**;  
 }  
}



## 2.2 BasePageResult

**public class** CmsPageResult **implements** Serializable{  
   
 **private static final long *serialVersionUID*** = 1L;  
   
 **private** Object **rows**;  
   
 **private** Integer **total**;  
   
 **public** CmsPageResult(Object rows, Integer total) {  
 **super**();  
 **this**.**rows** = rows;  
 **this**.**total** = total;  
 }  
  
 **public** Object getRows() {  
 **return rows**;  
 }  
  
 **public void** setRows(Object rows) {  
 **this**.**rows** = rows;  
 }  
  
 **public** Integer getTotal() {  
 **return total**;  
 }  
  
 **public void** setTotal(Integer total) {  
 **this**.**total** = total;  
 }  
}

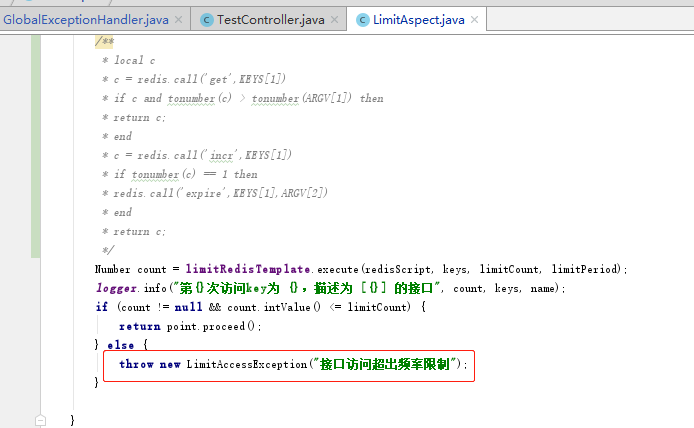




# 8 全局异常处理

## 8.1 @RestControllerAdvice

@RestControllerAdvice  
@Order(value = Ordered.***HIGHEST\_PRECEDENCE***)  
**public class** GlobalExceptionHandler {  
  
 @ExceptionHandler(value = AuthorizationException.**class**)  
 **public** Object handleAuthorizationException(HttpServletRequest request) {  
 **if** (*isAjaxRequest*(request)) {  
 **return** ResponseBo.*error*(**"暂无权限，请联系管理员！"**);  
 } **else** {  
 ModelAndView mav = **new** ModelAndView();  
 mav.setViewName(**"error/403"**);  
 **return** mav;  
 }  
 }  
  
 @ExceptionHandler(value = ExpiredSessionException.**class**)  
 **public** String handleExpiredSessionException() {  
 **return "login"**;  
 }  
  
 @ExceptionHandler(value = LimitAccessException.**class**)  
 **public** ResponseBo handleLimitAccessException(LimitAccessException e) {  
 **return** ResponseBo.*error*(e.getMessage());  
 }  
  
 @ExceptionHandler(value = ArithmeticException.**class**)  
 **public** ResponseBo testArithmeticException(ArithmeticException e) {  
 **return** ResponseBo.*error*(e.getMessage());  
 }  
  
 **private static boolean** isAjaxRequest(HttpServletRequest request) {  
 **return** (request.getHeader(**"X-Requested-With"**) != **null** && **"XMLHttpRequest"**.equals(request.getHeader(**"X-Requested-With"**)));  
 }  
}



## 8.2异常拦截器