# SpringBoot2学习笔记

## 静态资源目录

类路径下：called/static（or/public or /resources or /META-INF/resources）

访问：当前项目根路径/+静态资源名

原理：静态资源/\*\*

请求进来，先去找Controller看能不能处理，不能处理的所有请求又都交给静态资源处理器

静态资源能找到就访问，如果静态资源也找不到，就会报404错误

2）静态资源访问前缀

默认无前缀

|  |
| --- |
| spring:  mvc:  static-path-pattern: /res/\*\* |

访问路径：static-path-pattern+静态资源名=静态资源文件夹下面找

## 欢迎页支持

* 静态资源路径下：index.html

可以配置静态资源路径

但是不可以配置静态资源访问前缀，否则就会导致index.html不能被访问

|  |
| --- |
| spring: *# mvc: # static-path-pattern: /res/\*\** web:  resources:  static-locations: [classpath:/abc/] |

* Controller能处理/index

## 自定义Favicon

|  |
| --- |
| *#spring: # mvc: # static-path-pattern: /res/\*\* 这个也会影响favicon的功能 ### web: ## resources: ## static-locations: [classpath:/abc/]* |

## 请求参数处理

### 请求映射

1）@xxxMapping

2）Rest风格支持 （使用HTTP请求方式动词来表示对资源的操作）

以前：/getUser 获取用户 /deleteUser 删除用户 /editUser 修改用户 /saveUser 保存用户

现在 /user GET-获取用户 DELETE-删除用户 PUT-修改用户 POST-保存用户

核心Filter：HiddenHttpMethodFilter

用法：表单method=post，隐藏域\_method=put

SpringBoot中手动开启

|  |
| --- |
| mvc:  hiddenmethod:  filter:  enabled: true |

# Thymeleaf

## 基本语法：

|  |  |  |
| --- | --- | --- |
| 变量取值 | ${…} | 获取请求域、session、对象等 |
| 选择变量 | \*{……} | 获取上下文对象值 |
| 消息 | #{…} | 获取国际化等值 |
| 链接 | @{…} | 生成链接 |
| 片段表达式 | ~{…} | Jsp：include作用，引入公共页面 |

## 拦截器原理

1. 根据当前请求，找到handleExecutionChain[可以处理请求的handle以及handle的所有拦截器]
2. 先来顺序执行所有拦截器的prehandle方法
3. 如果当前拦截器的prehandle返回true。则执行下一个拦截器的prehandle  
   2.如果当前拦截器返回false。直接倒序执行所有已经执行了的拦截器的afterCompletion

3如果任何一个拦截器返回false。直接跳出不执行目标方法

4所有拦截器都返回true。执行目标方法

5.倒序执行所有拦截器的posthandle方法

6前面的步骤有任何异常都会直接触发afterCompletion

7.页面成功渲染以后，也会倒序执行触发afterCompletion

## 异常处理

### 错误处理

* 默认情况下，SpringBoot提供了/error处理所有错误的映射
* 对于机器客户端，它将生成JSON响应，其中包含错误，HTTP状态和异常信息的详细信息。对于浏览器客户端，响应一个“whitelabel”错误视图，以HTTP格式呈现相同的数据
* 要对其进行自定义，添加view解析为error
* 要完全替换默认行为，可以实现ErrorController并注册该类型的Bean定义，或添加ErrorAttribute类型的组件以实现现有机制但替换其内容

### 定制错误处理逻辑

* 自定义错误页
* @ControllerAdvice+@ExceptionHandler处理异常
* 实现HandlerExceptionResolver处理异常

### 异常处理自动配置原理

* ErrorMvcAutoConfiguration自动配置异常处理规则
* 容器中的组件：类型：DefaultErrorAttributes->id:errorAttributes
* Public class DefaultErrorAttributes implements ErrorAttributes,HandlerExceptionResolver
* DefaultErrorAttributes:定义错误页面中可以包含那些教程
* 容器中的组件：类型：BasicErrorController--🡪id:basicErrorController
* 处理默认/error路径的请求；页面响应new ModelAndView（“error”，moderl）
* 容器中有组件View 🡪id是error（响应默认错误页）
* 容器中组件BeanNameViewResovler（视图解析器）；按照返回的视图作为组件的id去容器中找View对象。
* 容器中的组件：类型：DefaultErrorViewResovler🡪id:conventionErrorViewResovler
* 如果发生错误，会以HTTP的状态码作为视图地址（ViewName），找到真正的页面
* Error/404,5xx.html

## Web原生组件注入（Servlet、Filter、Listener）

@ServletComponentScan(basePackages = "com.lyy.admin") ；指定原生Servlet组件都放在哪里

@WebServlet(urlPatterns = "/my") ；效果直接响应没有经过Spring的拦截器

## DispatcherServlet如何注册进来

* 容器中自动配置了DispatcherServlet属性绑定到WebMvcProperties；对应的配置文件满足spring.mvc
* 通过ServletRegistrationBean<DispatcherServlet>把DispatcherServlet配置进来
* 默认映射是”/”路径
* 多个Servlet都能处理到同一层路径，精确优先原则

## 数据访问·

* 引入mybatis-starter
* 配置application.yaml中,指定mapper-location位置即可
* 编写Mapper接口并标注@Mapper注解
* 简单方式直接注解方式
* 复杂方式编写mapper.xml进行绑定映射
* [//@MapperScan("com.lyy.admin.mapper")](mailto://@MapperScan(%22com.lyy.admin.mapper%22)) 其他的接口就不用标注@Mapper

## Junit5常用注解

Junit5的注解与Junit的注解有所变化

* @Test:表示方法是测试。但是与Junit4的@Test不同，他的职责非常单一不能声明任何属性，拓展的测试将会由Jupiter提供额外测试
* @ParameterizedTest：表示方法是参数化测试，下方会有详细介绍
* @RepeatedTest：表示方法可重复执行，下面会有介绍
* @DisplayName：为测试类或者测试方法设置展示名称
* @BeforeEach：表示在每个单元测试之前执行
* @AfterEach：表示在每个单元测试方法之后执行
* @BeforeAll：表示在所有单元测试之前执行
* @AfterAll：表示在所有单元测试之后执行
* @Tag：表示单元测试类别，类似于Junit4的@Categories
* @Disable：表示测试类或测试方法不执行，类似于Junit4的@Ignore
* @Timeout：表示测试方法运行如果超过了指定时间会将返回错误
* @ExtendWith：为测试类或测试方法提供扩展类引用

### 断言

检查业务逻辑返回的数据是否合理

1. 简单断言

|  |  |
| --- | --- |
| 方法 | 说明 |
| AssertEquals | 判断两个对象或两个原始类型是否相等 |
| AssertNotEquals | 判断两个对象或两个原始类型是都不相等 |
| AssertSame | 判断两个对象引用是否指向同一个对象 |
| AssertNotSame | 判断两个对象是否指向不同的对象 |
| AssertTrue | 判断给定的布尔值是否为true |
| AssertFalse | 判断给定的对象引用是否是为false |
| AssertNull | 判断给定的对象引用是否为Null |
| AssertNotNull | 判断给定的对象引用是否不为null |

### 前置条件

Junit5中的前置条件（assumptions【假设】）类似于断言，不同之处在于不满足的断言会使得测试方法失效，而不满足的前置条件只会使得测试方法的执行终止，前置条件可以看成是测试方法执行的前提，当该前提不满足时，就没有继承执行的必要

### 参数化测试

参数化测试是junit5很重要的一个新特性，他使得不同的参数多次运行测试成为了可能，也为我们的单元测试带来了许多便利

利用@ValueSource等注解，指定入参，我们将可以使用不同的参数进行多次单元测试，而不需要每新增一个参数就新增一个单元测试，省去了很多冗余代码

@ValueSource：为参数化测试指定入参来源，支持八大基础类以及String类型，Class类型

@NullSource：表示未参数化岑石提供一个null的入参

@EnumSource：表示未参数化测试提供一个枚举入参

@CsvFileSource：表示读取指定CSV文件内容作为参数化测试入参

@MethodSource：表示读取指定方法的返回值作为参数化测试入参（注意方法返需要是一个流）