5、Web-开发前期工作

2020年5月25日 11:11

- 1、简介

  使用SpringBoot；

  1）、创建SpringBoot应用，选中我们需要的模块；

  2）、SpringBoot已经默认将这些场景配置好了，只需要在配置文件中指定少量配置就可以运行起来

  3）、自己编写业务代码；

  自动配置原理?

  这个场景SpringBoot帮我们配置了什么？能不能修改？能修改哪些配置？能不能扩展？xxx

  xxxxAutoConfiguration：帮我们给容器中自动配置组件；
  xxxxProperties:配置类来封装配置文件的内容；

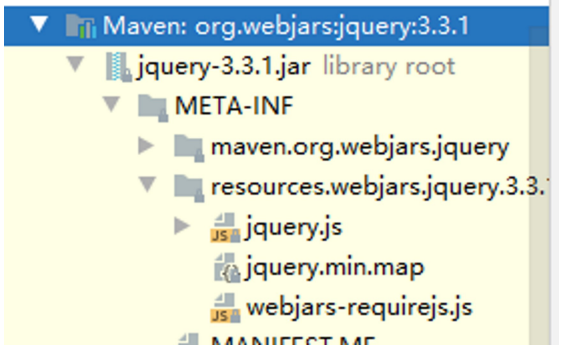
- 2、SpringBoot对静态资源的映射规则

```
@ConfigurationProperties(
        prefix="spring.resources",
        ignoreUnknownFields=false
)
publicclassResourceProperties{
//可以设置和静态资源有关的参数，缓存时间等
privatestaticfinalString[]CLASSPATH_RESOURCE_LOCATIONS=newString[]
{"classpath:/META-
INF/resources/","classpath:/resources/","classpath:/static/","classpath:/public/"};
//静态资源路径
publicvoidsetStaticLocations(String[]staticLocations){
this.staticLocations=this.appendSlashIfNecessary(staticLocations);
}
```

  - 1、所有 /webjars/** ，都去 classpath:/META-INF/resources/webjars/ 找资源；

    webjars：以jar包的方式引入静态资源；

    http://www.webjars.org/

    http://localhost:8080/webjars/jquery/3.3.1/jquery.js

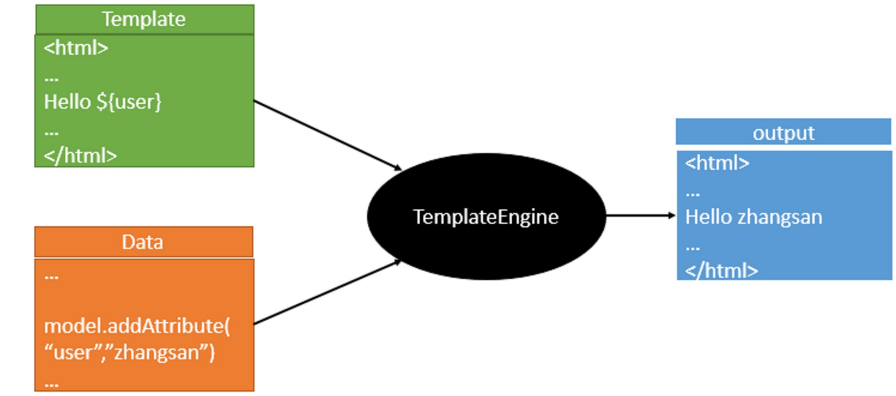
```
<!--引入jquery-webjar-->//在访问的时候只需要写webjars下面资源的名称即可
<dependency>
<groupId>org.webjars</groupId>
<artifactId>jquery</artifactId>
```

<version>3.3.1</version>
</dependency>

- 2、"/**" 访问当前项目的任何资源，都去（静态资源的文件夹）找映射
  "classpath:/META-INF/resources/",
  "classpath:/resources/",
  "classpath:/static/",
  "classpath:/public/"
  "/"：当前项目的根路径
  localhost:8080/abc ： 去静态资源文件夹里面找abc
- 3、欢迎页；静态资源文件夹下的所有index.html页面；被"/**"映射；==localhost:8080/ 找index页面
- 4、所有的 **/favicon.ico 都是在静态资源文件下找；（修改网页的图标）

- 3、模板引擎

  JSP、Velocity、Freemarker、Thymeleaf



  SpringBoot推荐的Thymeleaf；
  语法更简单，功能更强大；

  - 1、引入thymeleaf

```
<!--引入模板引擎-->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<!--切换thymeleaf版本-->
<properties>
<thymeleaf.version>3.0.9.RELEASE</thymeleaf.version>
<!-- 布局功能的支持程序 thymeleaf3主程序 layout2以上版本 -->
<!-- thymeleaf2   layout1-->
<thymeleaf-layout-dialect.version>2.2.2</thymeleaf-layout-dialect.version>
 </properties>
```

  - 2、thymeleaf使用

```
@ConfigurationProperties(
prefix="spring.thymeleaf"
)
publicclassThymeleafProperties{
privatestaticfinalCharsetDEFAULT_ENCODING;
publicstaticfinalStringDEFAULT_PREFIX="classpath:/templates/";
publicstaticfinalStringDEFAULT_SUFFIX=".html";
privatebooleancheckTemplate=true;
privatebooleancheckTemplateLocation=true;
```

```
privateStringprefix="classpath:/templates/";
privateStringsuffix=".html";
privateStringmode="HTML";
```

只要我们把HTML页面放在classpath:/templates/，thymeleaf就能自动渲染；

使用：

- 1、导入thymeleaf的名称空间

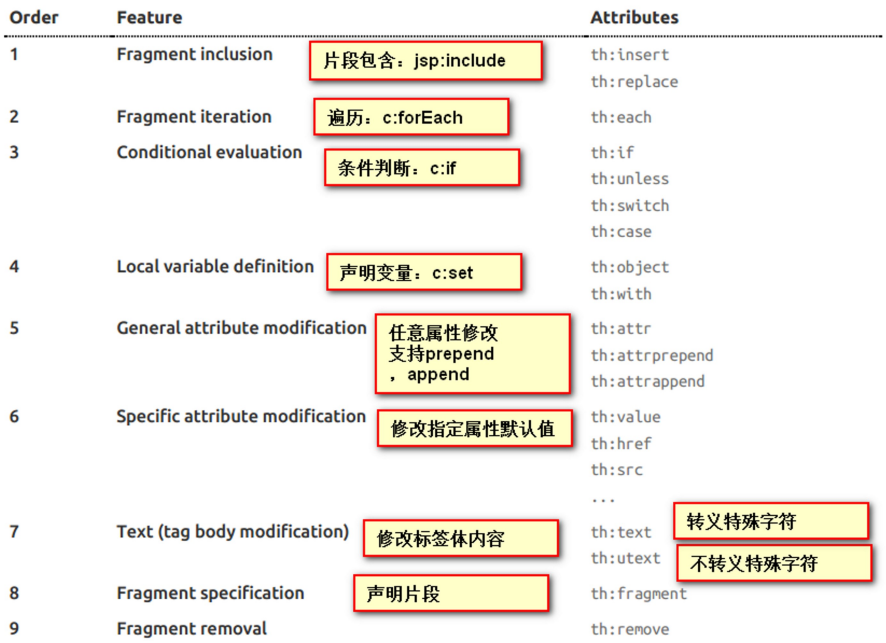
  ```
  <html lang="en" xmlns:th="http://www.thymeleaf.org">
  ```

- 2、使用thymeleaf语法；

  ```
  <div th:text="${hello}"></div>
  ```

- 3、语法规则
  - 1、th:text；改变当前元素里面的文本内容；

    th：任意html属性；来替换原生属性的值



  - 2、表达式

    Simple expressions:（表达式语法）

    Variable Expressions: ${...}：获取变量值；OGNL；

    1）、获取对象的属性、调用方法

    2）、使用内置的基本对象：

    ```
    #ctx : the context object.
    #vars: the context variables.
    #locale : the context locale.
    #request : (only in Web Contexts) the HttpServletRequest object.
    #response : (only in Web Contexts) the HttpServletResponse object.
    #session : (only in Web Contexts) the HttpSession object.
    #servletContext : (only in Web Contexts) the ServletContext object.
    ${session.foo}
    ```

    3）、内置的一些工具对象：

    ```
    #execInfo : information about the template being processed.
    #messages : methods for obtaining externalized messages inside variables
    expressions, in the same way as they would be obtained using #{…} syntax.
    #uris : methods for escaping parts of URLs/URIs
    #conversions : methods for executing the configured conversion service (if any).
    #dates : methods for java.util.Date objects: formatting, component extraction,
    etc.
    ```

#calendars : analogous to #dates , but for java.util.Calendar objects.
#numbers : methods for formatting numeric objects.
#strings : methods for String objects: contains, startsWith, prepending/appending, etc.
#objects : methods for objects in general.
#bools : methods for boolean evaluation.
#arrays : methods for arrays.
#lists : methods for lists.
#sets : methods for sets.
#maps : methods for maps.
#aggregates : methods for creating aggregates on arrays or collections.
#ids : methods for dealing with id attributes that might be repeated (for example, as a result of an iteration).

Selection Variable Expressions: *{...}：选择表达式：和${}在功能上是一样；
补充：配合 th:object="${session.user}：

Name: Sebastian.
Surname: Pepper.
Nationality: Saturn.

Message Expressions: #{...}：获取国际化内容
Link URL Expressions: @{...}：定义URL；
@{/order/process(execId=${execId},execType='FAST')}
Fragment Expressions: ~{...}：片段引用表达式
...

Literals（字面量）
Text literals: 'one text' , 'Another one!' ,...
Number literals: 0 , 34 , 3.0 , 12.3 ,...
Boolean literals: true , false
Null literal: null
Literal tokens: one , sometext , main ,...
Text operations:（文本操作）
String concatenation: +
Literal substitutions: |The name is ${name}|
Arithmetic operations:（数学运算）
Binary operators: + , - , * , / , %
Minus sign (unary operator): -
Boolean operations:（布尔运算）
Binary operators: and , or
Boolean negation (unary operator): ! , not
Comparisons and equality:（比较运算）
Comparators: > , < , >= , <= (gt , lt , ge , le)
Equality operators: == , != (eq , ne)
Conditional operators:条件运算（三元运算符）
If-then: (if) ? (then)
If-then-else: (if) ? (then) : (else)
Default: (value) ?: (defaultvalue)
Special tokens:
No-Operation: _

- 4、SpringMVC自动配置
  https://docs.spring.io/spring-boot/docs/2.2.7.RELEASE/reference/htmlsingle/#boot-features-developing-web-applications

- 1、Spring MVC auto-configuration
  Spring Boot 自动配置好了SpringMVC
  以下是SpringBoot对SpringMVC的默认配置:**(WebMvcAutoConfiguration)**
  - Inclusion
    of ContentNegotiatingViewResolver and BeanNameViewResolver beans.
    - 自动配置了ViewResolver（视图解析器：根据方法的返回值得到视图对象（View），视图对象决定如何渲染（转发？重定向？））
    - ContentNegotiatingViewResolver：组合所有的视图解析器的；
    - 如何定制：我们可以自己给容器中添加一个视图解析器；自动的将其组合进来；
  - Support for serving static resources, including support for WebJars (covered later in this document)).
    - 静态资源文件夹路径,webjars
  - Automatic registration of Converter, GenericConverter, and Formatter beans.
    - Converter：转换器； public String hello(User user)：类型转换使用Converter
    - Formatter 格式化器； 2020.05.25===Date;
      ```
      @Bean
      @ConditionalOnProperty(prefix = "spring.mvc", name = "date-format")//在文件中配置日期格式化的规则
      public Formatter<Date> dateFormatter() {
          return new DateFormatter(this.mvcProperties.getDateFormat());//日期格式化组件
      }
      ```
    - 自己添加的格式化器转换器，我们只需要放在容器中即可
  - Support for HttpMessageConverters (covered later in this document).
    - HttpMessageConverter：SpringMVC用来转换Http请求和响应的；User---Json；
    - HttpMessageConverter是从容器中确定；获取所有的HttpMessageConverter；
    - 自己给容器中添加HttpMessageConverter，只需要将自己的组件注册容器中（@Bean,@Component）
  - Automatic registration of MessageCodesResolver (covered later in this document).定义错误代码生成规则
  - Static index.html support.静态首页访问
  - Custom Favicon support (covered later in this document).favicon.ico
  - Automatic use of a ConfigurableWebBindingInitializer bean
    (covered later in this document).我们可以配置一个ConfigurableWebBindingInitializer来替换默认的；（添加到容器）
    初始化WebDataBinder；
    请求数据=====JavaBean；
    org.springframework.boot.autoconfigure.web：web的所有自动场景；

- 2、扩展SpringMVC
  - 编写一个配置类（@Configuration），实现WebMvcConfigurer接口；不能标注@EnableWebMvc

    既保留了所有的自动配置，也能用我们扩展的配置；

```
//使用WebMvcConfigurer可以来扩展SpringMVC的功能
@Configuration
publicclassMyMvcConfigimplementsWebMvcConfigurer{
@Override
publicvoidaddViewControllers(ViewControllerRegistryregistry){
//浏览器发送/atLhq请求来到success
registry.addViewController("/atLhq").setViewName("success");
}
}
```

  - 原理：
    - 1、WebMvcAutoConfiguration是SpringMVC的自动配置类
    - 2、在做其他自动配置时会导入；@Import(**EnableWebMvcConfiguration**.class)
    - 3、容器中所有的WebMvcConfigurer都会一起起作用；
    - 4、我们的配置类也会被调用；

      效果：SpringMVC的自动配置和我们的扩展配置都会起作用；
- 3、全面接管SpringMVC；

  SpringBoot对SpringMVC的自动配置不需要了，所有都是我们自己配置；所有的SpringMVC的自动配置都失效了

  我们需要在配置类中添加@EnableWebMvc即可；

```
@EnableWebMvc
@Configuration
publicclassMyMvcConfigimplementsWebMvcConfigurer{
@Override
publicvoidaddViewControllers(ViewControllerRegistryregistry){
//浏览器发送/atLhq请求来到success
registry.addViewController("/atLhq").setViewName("success");
}
}
```

- 5、如何修改SpringBoot的默认配置

  模式：
  - 1、SpringBoot在自动配置很多组件的时候，先看容器中有没有用户自己配置的（@Bean、@Component）如果有就用用户配置的，如果没有，才自动配置；如果有些组件可以有多个（ViewResolver）将用户配置的和自己默认的组合起来；
  - 2、在SpringBoot中会有非常多的xxxConfigurer帮助我们进行扩展配置
  - 3、在SpringBoot中会有很多的xxxCustomizer帮助我们进行定制配置