链滴

SSM 框架—Thymeleaf 模板引擎 Spring5 整合 Thymeleaf（XML 配置 和 注解配置）

作者：ilssio

原文链接：https://ld246.com/article/1558615679341

来源网站：链滴

1. 依赖

- 在配置好SSM框架后，在**pom.xml**中添加如下依赖

```
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf-spring5</artifactId>
    <version>3.0.9.RELEASE</version>
</dependency>
```

2. 配置

- **配置文件方式**

- 在**Sping_mvc.xml**中加入如下配置，并且注释掉jsp的viewResolver或freemarker的配置

```
<bean id="templateResolver"
      class="org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver">
    <property name="prefix" value="/WEB-INF/templates/"/>
    <property name="suffix" value=".html"/>
    <property name="characterEncoding" value="UTF-8"/>
    <property name="order" value="1"/>
    <property name="templateMode" value="HTML5"/>
    <property name="cacheable" value="false"/>
</bean>

<bean id="templateEngine"
      class="org.thymeleaf.spring5.SpringTemplateEngine">
    <property name="templateResolver" ref="templateResolver"/>
</bean>

<bean id="viewResolver" class="org.thymeleaf.spring5.view.ThymeleafViewResolver">
    <property name="templateEngine" ref="templateEngine"/>
    <property name="characterEncoding" value="UTF-8"/>
</bean>
```

- **注解配置**

在**webConfig.java中**中加入如下配置，如果配置了jsp或者freemarker请注释掉jsp的viewResolver或reemarker的配置

```
@Configuration
@EnableWebMvc
@ComponentScan({"com.example.controller", "com.example.api"})
public class WebConfig implements WebMvcConfigurer {

    /**
     * 模板解析器
     *
     * @return
     */
    @Bean
    public SpringResourceTemplateResolver templateResolver() {
```

```
        SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateReso
ver();
        templateResolver.setPrefix(TEMPLATE_PREFIX);
        templateResolver.setSuffix(TEMPLATE_SUFFIX);
        templateResolver.setCacheable(TEMPLATE_CACHEABLE);
        templateResolver.setCharacterEncoding(CHARACTER_ENCODING);
        templateResolver.setTemplateMode(TEMPLATE_MODE);
        templateResolver.setOrder(TEMPLATE_ORDER);
        return templateResolver;
    }

    /**
     * 模板引擎
     *
     * @return
     */
    @Bean
    public SpringTemplateEngine springTemplateEngine(SpringResourceTemplateResolver te
plateResolver) {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver);
        return templateEngine;
    }

    /**
     * 视图解析器
     *
     * @return
     */
    @Bean
    public ThymeleafViewResolver viewResolver(SpringTemplateEngine springTemplateEngine)

        ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
        viewResolver.setTemplateEngine(springTemplateEngine);
        viewResolver.setCharacterEncoding(CHARACTER_ENCODING);
        return viewResolver;
    }
//.....其他配置请自行配置
}
```

由于我使用了CONSTANTS类里面的static变量，所以附上Constants类相关的参数

```
    public final static String CHARACTER_ENCODING = "UTF-8";

    /**
     * thymeleaf模板引擎参数
     */
    public final static String TEMPLATE_PREFIX = "/WEB-INF/templates/";
    public final static String TEMPLATE_SUFFIX = ".html";
    public final static Boolean TEMPLATE_CACHEABLE = false;
    public final static String TEMPLATE_MODE = "HTML5";
    public final static Integer TEMPLATE_ORDER = 1;
```

此配置需要注意以下几点：

- templateResolver的prefix与suffix对应你的视图层的文件位置
- templateResolver的characterEncoding和viewResolver的都要设置成UTF-8中文才不会乱码。
- templateResolver的cacheable一定要在开发的时候设置成false不然无法看到实时的页面数据

3. 测试

- controller：

在ModelMap里面随便设置一点值

```
@RequestMapping("/test")
public String test(ModelMap map) {
    map.put("thText", "设置文本内容");
    map.put("thUText", "设置文本内容");
    map.put("thValue", "设置当前元素的value值");
    map.put("thEach", Arrays.asList("列表", "遍历列表"));
    map.put("thIf", "msg is not null");
    map.put("thObject", new                    UserEntity("sadfa","asfasfd","asfsaf","asdfasf","sa
","asfd","sadf",1));

    return "test";
}
```

- test.html

```
<!DOCTYPE html>
<html lang="cn" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>TEST</h1>
<h2>Thymeleaf</h2>
<!--th:text 设置当前元素的文本内容，常用，优先级不高-->
<p th:text="${thText}" />
<p th:utext="${thUText}" />

<!--th:value 设置当前元素的value值，常用，优先级仅比th:text高-->
<input type="text" th:value="${thValue}" />

<!--th:each 遍历列表，常用，优先级很高，仅此于代码块的插入-->
<!--th:each 修饰在div上，则div层重复出现，若只想p标签遍历，则修饰在p标签上-->
<div th:each="message : ${thEach}"> <!-- 遍历整个div-p，不推荐-->
    <p th:text="${message}" />
</div>
<div> <!--只遍历p，推荐使用-->
    <p th:text="${message}" th:each="message : ${thEach}" />
</div>

<!--th:if 条件判断，类似的有th:switch，th:case，优先级仅次于th:each, 其中#strings是变量表达
的内置方法-->
<p th:text="${thIf}" th:if="${not #strings.isEmpty(thIf)}"></p>
```

```
<!--th:insert 把代码块插入当前div中，优先级最高，类似的有th:replace，th:include，~{}：代码表达式 -->
<div th:insert="~{grammar/common::thCommon}"></div>

<!--th:object 声明变量，和*{} 一起使用-->
<div th:object="${thObject}">
    <p>ID: <span th:text="*{id}" /></p><!--th:text="${thObject.id}"-->
    <p>TH: <span th:text="*{username}" /></p><!--${thObject.thName}-->
    <p>DE: <span th:text="*{password}" /></p><!--${thObject.desc}-->
</div>
</body>
</html>
```

几点注意：

- 在html首标签里面加上xmlns

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

- 同样把head的meta设置一个charset=“UTF-8”

至此，你就可以去配置tomcat运行项目了。