第 3 章： SpringBoot 整合视图层

1.1 本章大纲

- SpringBoot 整合 JSP
- SpringBoot 整合 freemarker
- SpringBoot 整合 Thymeleaf

1.2 SpringBoot 整合 JSP

1.2.1 在 pom.xml 文件添加 jsp 相关 jar 依赖

```
<!-- jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
<!-- jasper -->
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```

1.2.2 编写 application.properties 文件

```
#页面访问前缀路径
spring.mvc.view.prefix=/WEB-INF/jsp/
#页面访问后缀
spring.mvc.view.suffix=.jsp
```

1.2.3 创建 Controller

```
@Controller
public class UserController {

    /*
     * 处理请求，产生数据
     */
    @RequestMapping("/showUser")
    public String showUser(Model model){
        List<Users> list = new ArrayList<>();
        list.add(new Users(1,"张三",20));
        list.add(new Users(2,"李四",22));
        list.add(new Users(3,"王五",24));

        //需要一个Model对象
        model.addAttribute("list", list);
        //跳转视图
        return "userList";
    }
}
```

1.2.4 创建 jsp 页面

```
<body>
    <table border="1" align="center" width="50%">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Age</th>
        </tr>
        <c:forEach items="${list }" var="user">
            <tr>
                <td>${user.userid }</td>
                <td>${user.username }</td>
                <td>${user.userage }</td>
            </tr>
        </c:forEach>
    </table>
</body>
```

1.3 SpringBoot 整合 freemarker

1.3.1 在 pom.xml 文件添加 freemarker 相关 jar 依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <!-- freemarker 启动器的坐标 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-freemarker</artifactId>
    </dependency>
</dependencies>
```

1.3.2 创建 Controller

```
@RequestMapping("/showUser")
public String showUser(Model model){
    List<Users> list = new ArrayList<>();
    list.add(new Users(1,"张三",20));
    list.add(new Users(2,"李四",22));
    list.add(new Users(3,"王五",24));
    list.add(new Users(3,"王五",24));
    //将数据放到模型组
    model.addAttribute("list", list);
    //跳转视图
    return "userList";
}
```

1.3.3 创建 freemarker 模板页面

注意：springBoot 要求模板形式的视图层技术的文件必须要放到 src/main/resources 目录下必须要一个名称为 templates

Freemarker 文件后缀名是.ftl

```
<html>
    <head>
        <title>展示用户数据</title>
        <meta charset="utf-9"></meta>
    </head>
    <body>
        <table border="1" align="center" width="50%">
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Age</th>
            </tr>
            <#list list as user >
                <tr>
                    <td>${user.userid}</td>
                    <td>${user.username}</td>
                    <td>${user.userage}</td>
                </tr>
            </#list>
        </table>
    </body>
</html>
```

1.4 SpringBoot 整合 Thymeleaf

1.4.1 创建 Thymeleaf 入门项目

1.4.1.1 在 pom.xml 文件添加 thymeleaf 依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
```

1.4.1.2 编写 application.properties 文件

```
#thymelea模板配置
spring.thymeleaf.prefix=classpath:/templates/
#thymeleaf文件后缀
spring.thymeleaf.suffix=.html
#模式
spring.thymeleaf.mode=HTML5
#编码格式
spring.thymeleaf.encoding=UTF-8
```

1.4.1.3 Thymeleaf 视图存放目录

目录位置：src/main/resources/templates
templates：该目录是安全的。意味着该目录下的内容是不允许外界直接访问的。

1.4.2 Thymeleaf 的基本使用

1.4.2.1 Thymeleaf 特点

Thymelaef 是通过特定语法对 html 的标记做渲染

1.4.2.2 创建 Controller

```
@Controller
public class UserController {

    @RequestMapping("/show")
    public String show(Model model) {
        System.out.println("进入控制器....");
        model.addAttribute("msg", "Thymeleaf案例");
        model.addAttribute("dateKey", new Date());
        model.addAttribute("sex", "男");
        model.addAttribute("id", 2);
        return "test";
    }
```

1.4.2.3 创建视图 html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
```

1.4.2.4 注意 thymeleaf 版本

注意 thymeleaf 版本问题，如果版本是 3.0 以下，则需要在 pom.xml 文件加上以下配置：

```
<properties>
    <java.version>1.8</java.version>
    <thymeleaf.version>3.0.9.RELEASE</thymeleaf.version>
    <thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.version>
</properties>
```

建议使用 3.0 以上版本

1.4.3 Thymeleaf 语法

1.4.3.1 变量输出与字符串操作

1.4.3.1.1 th:text

作用：在页面中输出值

```
<span th:text="hello"></span>
```

1.4.3.1.2 th:value

作用：可以将一个值放入到 input 标签的 value 中

```
<input th:value="${msg}">
```

1.4.3.2 判断字符串是否为空

Thymeleaf 内置对象

注意语法：

1，调用内置对象一定要用#

2，大部分的内置对象都以 s 结尾 strings、numbers、dates

${#strings.isEmpty(key)}
判断字符串是否为空，如果为空返回 true，否则返回 false
${#strings.contains(msg,'T')}

判断字符串是否包含指定的字符串，如果包含返回 true，否则返回 false
${#strings.startsWith(msg,'a')}
判断当前字符串是否以字符串开头，如果是返回 true，否则返回 false
${#strings.endsWith(msg,'a')}
判断当前字符串是否以字符串结尾，如果是返回 true，否则返回 false
${#strings.length(msg)}
返回字符串的长度
${#strings.indexOf(msg,'h')}
查找字符串的位置，并返回该字符串的下标，如果没找到则返回-1
${#strings.substring(msg,13)} ${#strings.substring(msg,13,15)}
截取字符串，用法与 String 类下 SubString() 方法相同
${#strings.toUpperCase(msg)} ${#strings.toLowerCase(msg)}
字符串转大小写。

1.4.3.3 日期格式处理

${#dates.format(key)}
格式化日期，默认的以浏览器默认语言为格式化标准
${#dates.format(key,'yyy/MM/dd')}
按照自定义的格式做日期转换
${#dates.year(key)} ${#dates.month(key)} ${#dates.day(key)}
year：取年 Month：取月 Day：取日

1.4.3.4 条件判断

1.4.3.4.1 th:if

```
<span th:if="${sex} == ' 男 '">
    性别：男
</span>
<span th:if="${sex} == ' 女 '">
    性别：女
</span>
```

1.4.3.4.2 th:switch

```
<div th:switch="${id}">
    <span th:case="1">1</span>
    <span th:case="2">2</span>
    <span th:case="3">3</span>
</div>
```

1.4.3.5 迭代循环

1.4.3.5.1 th:each

1.4.3.5.1.1控制器

```
/*
 * 处理请求，产生数据
 */
@RequestMapping("/showUser")
public String showUser(Model model){
    List<Users> list = new ArrayList<>();
    list.add(new Users(1,"张三",20));
    list.add(new Users(2,"李四",22));
    list.add(new Users(3,"王五",24));
    list.add(new Users(3,"王五",24));
    //需要一个Model对象
    model.addAttribute("list", list);
    //跳转视图
    return "userList";
}
```

1.4.3.5.1.2页面

```
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr th:each="u : ${list}">
        <td th:text="${u.userid}"></td>
        <td th:text="${u.username}"></td>
        <td th:text="${u.userage}"></td>
    </tr>
</table>
```

1.4.3.5.2 th:each 状态变量

```
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Index</th>
        <th>Count</th>
        <th>Size</th>
        <th>Even</th>
        <th>Odd</th>
        <th>First</th>
        <th>lase</th>
    </tr>
    <tr th:each="u,var : ${list}">
        <td th:text="${u.userid}"></td>
        <td th:text="${u.username}"></td>
        <td th:text="${u.userage}"></td>
        <td th:text="${var.index}"></td>
        <td th:text="${var.count}"></td>
        <td th:text="${var.size}"></td>
        <td th:text="${var.even}"></td>
        <td th:text="${var.odd}"></td>
        <td th:text="${var.first}"></td>
        <td th:text="${var.last}"></td>
    </tr>
</table>
```

状态变量属性

1,index:当前迭代器的索引 从 0 开始

2,count:当前迭代对象的计数 从 1 开始

3,size:被迭代对象的长度

4,even/odd:布尔值，当前循环是否是偶数/奇数 从 0 开始

5,first:布尔值，当前循环的是否是第一条，如果是返回 true 否则返回 false

6,last:布尔值，当前循环的是否是最后一条，如果是则返回 true 否则返回 false

1.4.3.5.3 th:each 迭代 map

1.4.3.5.3.1控制器

```
/*
 * 处理请求，产生数据
 */
@RequestMapping("/showUser2")
public String showUser2(Model model){
    Map<String, Users> map = new HashMap<>();
    map.put("u1", new Users(1,"张三",20));
    map.put("u2", new Users(2,"李四",22));
    map.put("u3", new Users(3,"王五",24));
    //将数据添加到模型中
    model.addAttribute("map", map);
    //跳转视图
    return "userList2";
}
```

1.4.3.5.3.2页面

```
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr th:each="maps: ${map}">
        <td th:each="entry:${maps}" th:text="${entry.value.userid}"></td>
        <td th:each="entry:${maps}" th:text="${entry.value.username}"></td>
        <td th:each="entry:${maps}" th:text="${entry.value.userage}"></td>
    </tr>
</table>
```

1.4.3.6 域对象操作

1.4.3.6.1 HttpServletRequest

1.4.3.6.1.1控制器

```
request.setAttribute("req", "张三");
```

1.4.3.6.1.2页面

```
<h2>request：<span th:text="${req}"></span></h2>
<h2>request：<span th:text="${#httpServletRequest.getAttribute('req')}"></span></h2>
```

1.4.3.6.2 HttpSession

1.4.3.6.2.1控制器

```
request.getSession().setAttribute("ses", "李四");
```

1.4.3.6.2.2页面

```
<h2>session: <span th:text="${session.ses}"></span></h2>
<h2>session: <span th:text="${#httpSession.getAttribute('ses')}"></span></h2>
```

1.4.3.6.3 ServletContext

1.4.3.6.3.1控制器

```
request.getSession().getServletContext().setAttribute("app", "王五");
```

1.4.3.6.3.2页面

```
<h2>application: <span th:text="${application.app}"></span></h2>
```

1.4.3.7 URL 表达式

th:href
th:src

1.4.3.7.1 url 表达式语法

基本语法：@{}

1.4.3.7.2 url 类型

1.4.3.7.2.1绝对路径

```
<a th:href="@{http://www.baidu.com}">绝对路径</a>
```

1.4.3.7.2.2相对路径

1)相对于当前项目的根
相对于项目的上下文的相对路径

```
<a th:href="@{/attr}">相对路径</a>
```

2) 相对于服务器路径的根

```
<a th:href="@{~/project2/resourcename}">相对于服务器的根</a>
```

1.4.3.8 在 url 中实现参数传递

```
<a th:href="@{/attr(id=1,name=zhagnsan)}">相对路径-传参</a>
```

1.4.3.9 在 url 中通过 restful 风格进行参数传递

```
<a th:href="@{/path/{id}/show(id=1,name=zhagnsan)}"> 相对路径 - 传参-restful</a>
```