

## Spring Boot 第三章（视图层技术）

### 课程介绍:

- 整合 jsp
- 整合 freemarker
- 整合 Thymeleaf

### 一，SpringBoot 整合 jsp 技术

#### 1,创建项目

New Maven Project

Configure project

Artifact

Group Id: com.bjsxt

Artifact Id: 08-spring-boot-view-jsp

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id: org.springframework.boot

Artifact Id: spring-boot-starter-parent

Version: 1.5.10.RELEASE

Browse... Clear

Advanced

< Back Next > Finish Cancel

## 2,修改 pom 文件，添加坐标

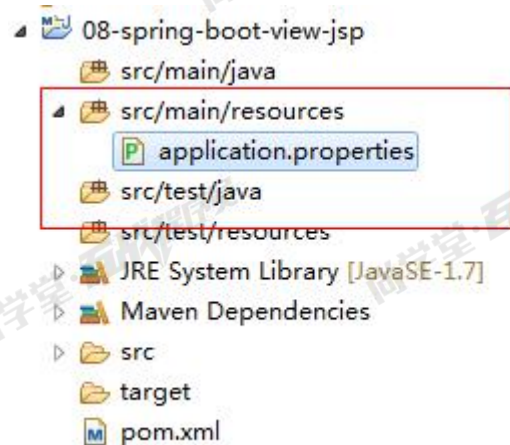
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.10.RELEASE</version>
  </parent>
  <groupId>com.bjsxt</groupId>
  <artifactId>08-spring-boot-view-jsp</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <!-- jdk1.7 -->
  <properties>
    <java.version>1.7</java.version>
  </properties>

  <dependencies>
    <!-- springBoot 的启动器 -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!-- 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>
  </dependencies>
</project>
```

### 3,创建 springBoot 的全局配置文件， application.properties

```
spring.mvc.view.prefix=/WEB-INF/jsp/  
spring.mvc.view.suffix=.jsp
```

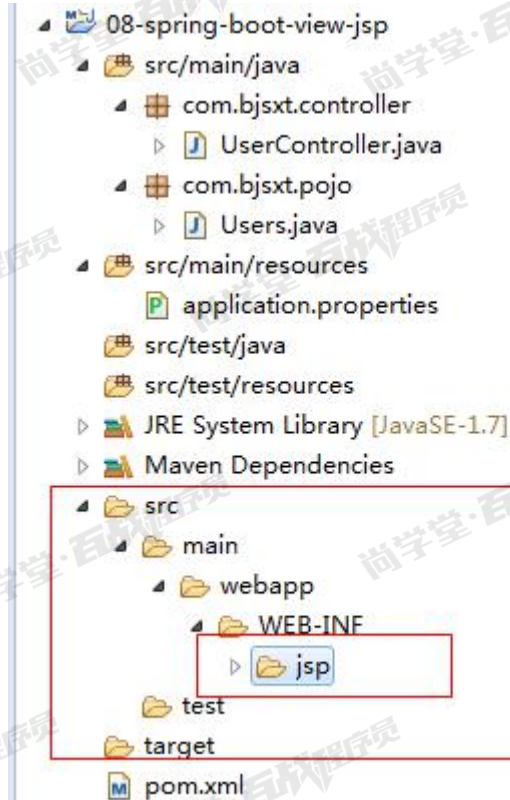


### 4,创建 Controller

```
/**  
 * SpringBoot 整合 jsp  
 *  
 */  
@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";  
    }  
}
```

```
}  
}
```

## 5,创建 jsp



```
<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Insert title here</title>  
</head>  
<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>
</html>
```

## 6,创建启动类

```
/**
 * SpringBoot 启动类
 *
 */
@SpringBootApplication
public class App {

    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }
}
```



## 二, SpringBoot 整合 Freemarker

### 1, 创建项目

New Maven Project

Configure project

Artifact

Group Id: com.bjsxt

Artifact Id: 09-spring-boot-view-freemarker

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id: org.springframework.boot

Artifact Id: spring-boot-starter-parent

Version: 1.5.10.RELEASE

Browse... Clear

Advanced

< Back Next > Finish Cancel

### 2, 修改 pom 添加坐标

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.10.RELEASE</version>
  </parent>
  <groupId>com.bjsxt</groupId>
```

```

<artifactId>09-spring-boot-view-freemarker</artifactId>
<version>0.0.1-SNAPSHOT</version>

<properties>
  <java.version>1.7</java.version>
</properties>

<dependencies>
<!-- springBoot 的启动器 -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <!-- freemarker 启动器的坐标 -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
  </dependency>
</dependencies>
</project>

```

### 3,编写视图

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

```

<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>

```

## 4,创建 Controller

```

/**
 * SpringBoot 整合 jsp
 *
 *
 */
@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";
    }
}

```

## 5,创建启动器

```

/**
 * SpringBoot 启动类
 *

```



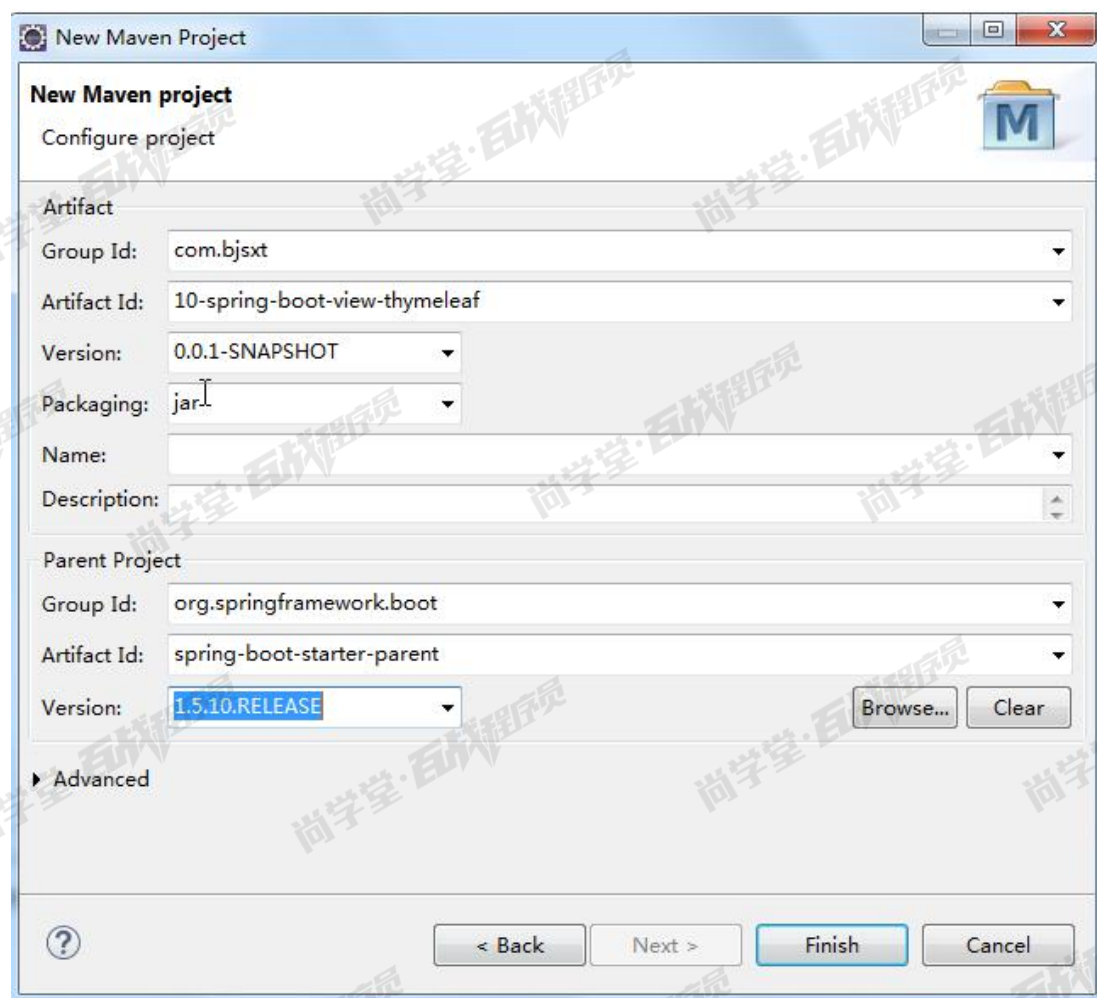
```
*/
*/
@SpringBootApplication
public class App {

    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }
}
```

### 三，SpringBoot 整合 Thymeleaf （重点讲解）

#### 1. 创建 Thymeleaf 的入门项目

##### 1.1 创建项目



New Maven Project

Configure project

Artifact

Group Id: com.bjsxt

Artifact Id: 10-spring-boot-view-thymeleaf

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id: org.springframework.boot

Artifact Id: spring-boot-starter-parent

Version: 1.5.10.RELEASE

Browse... Clear

Advanced

< Back Next > Finish Cancel

##### 1.2 修改 pom 文件添加坐标

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.5.10.RELEASE</version>
</parent>
<groupId>com.bjsxt</groupId>
<artifactId>10-spring-boot-view-thymeleaf</artifactId>
<version>0.0.1-SNAPSHOT</version>

<properties>
<java.version>1.7</java.version>
</properties>

<dependencies>
<!-- springBoot 的启动器 -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- springBoot 的启动器 -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
</dependencies>
</project>
```

## 1.3 创建存放视图的目录

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。

## 2. Thymeleaf 的基本使用

### 2.1 Thymeleaf 特点：

Thymeleaf 是通过他特定语法对 html 的标记做渲染。

## 2.2 编写 Controller

```
/**
 * Thymeleaf 入门案例
 *
 */
@Controller
public class DemoController {
    @RequestMapping("/show")
    public String showInfo(Model model){
        model.addAttribute("msg", "Thymeleaf 第一个案例");
        return "index";
    }
}
```

## 2.3 创建视图 .html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Thymeleaf 入门</title>
</head>
<body>
    <span th:text="Hello"></span>
    <hr/>
    <span th:text="${msg}"></span>
</body>
</html>
```

## 2.4 编写启动类

```
/**
 *
 * Thymeleaf 入门案例
 *
 */
@SpringBootApplication
public class App {
```

```

public static void main(String[] args) {
    SpringApplication.run(App.class, args);
}
}

```

## 2.5 解决异常

[org.xml.sax.SAXParseException](#): 元素类型 "meta" 必须由匹配的结束标记 "</meta>" 终止。

```

at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.createSAXParseException(
at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.fatalError(
at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(
at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(
at com.sun.org.apache.xerces.internal.impl.XMLScanner.reportFatalError(
at com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.sc

```

### 2.5.1 解决异常方式 1

让 html 的标记按照严禁的语法去编写。

```

<meta charset="UTF-8"/>
<title>Thymeleaf 入门</title>

```

### 2.5.2 解决异常方式 2

Thymeleaf.jar: 更新为 3.0 以上的版本

thymeleaf-layout-dialect.jar: 更新为 2.0 以上的版本

更换 thymeleaf 的 jar 包的版本

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.10.RELEASE</version>
    </parent>
    <groupId>com.bjsxt</groupId>
    <artifactId>10-spring-boot-view-thymeleaf</artifactId>
    <version>0.0.1-SNAPSHOT</version>

```

```

<properties>
<java.version>1.7</java.version>
<thymeleaf.version>3.0.2.RELEASE</thymeleaf.version>
<thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.version>
</properties>

<dependencies>
<!-- springBoot 的启动器 -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- springBoot 的启动器 -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
</dependencies>
</project>

```

### 3. Thymeleaf 语法详解

#### 3.1 变量输出与字符串操作

##### 3.1.1th:text

th:text
---------

在页面中输出值
---------

##### 3.1.2th:value

th:value
----------

可以将一个值放入到 input 标签的 value 中
-----------------------------



### 3.1.3 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

<code>#{@strings.isEmpty(key)}</code>
判断字符串是否为空，如果为空返回 true，否则返回 false
<code>#{@strings.contains(msg, 'T')}</code>
判断字符串是否包含指定的子串，如果包含返回 true，否则返回 false
<code>#{@strings.startsWith(msg, 'a')}</code>
判断当前字符串是否以子串开头，如果是返回 true，否则返回 false
<code>#{@strings.endsWith(msg, 'a')}</code>
判断当前字符串是否以子串结尾，如果是返回 true，否则返回 false
<code>#{@strings.Length(msg)}</code>
返回字符串的长度
<code>#{@strings.indexOf(msg, 'h')}</code>
查找子串的位置，并返回该子串的下标，如果没找到则返回-1
<code>#{@strings.substring(msg, 13)}</code> <code>#{@strings.substring(msg, 13, 15)}</code>
截取子串，用户与 jdk String 类下 SubString 方法相同
<code>#{@strings.toUpperCase(msg)}</code> <code>#{@strings.toLowerCase(msg)}</code>
字符串转大小写。

### 3.2 日期格式化处理

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

### 3.3 条件判断

#### 3.3.1th:if

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

#### 3.3.2th:switch

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

```
<span th:case="3">ID 为 3</span>
</div>
```

## 3.4 迭代遍历

### 3.4.1th:each

```
@RequestMapping("/show3")
public String showInfo3(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));
    model.addAttribute("list", list);
    return "index3";
}

<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>
```

### 3.4.2th:each 状态变量

```
@RequestMapping("/show3")
public String showInfo3(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));
    model.addAttribute("list", list);
    return "index3";
}
```

```

    }
    <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>Last</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

### 3.4.3th:each 迭代 Map

```

@RequestMapping("/show4")
public String showInfo4(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));
        model.addAttribute("map", map);
        return "index4";
    }

```

```

<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr th:each="maps : ${map}">
        <td th:text="${maps}"></td>
    </tr>
</table>
<th/>
<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>

```

## 3.5 域对象操作

### 3.5.1 HttpServletRequest

```

request.setAttribute("req", "HttpServletRequest");
Request:<span
th:text="${#httpServletRequest.getAttribute('req')}"></span><br/>

```

### 3.5.2 HttpSession

```
request.getSession().setAttribute("sess", "HttpSession");  
Session:<span th:text="${session.sess}"></span><br/>
```

### 3.5.3 ServletContext

```
request.getSession().getServletContext().setAttribute("app",  
"Application");  
Application:<span th:text="${application.app}"></span>
```

## 3.6 URL 表达式

th:href  
th:src

### 3.6.1 url 表达式语法

基本语法: @{ }

### 3.6.2 URL 类型

#### 3.6.2.1 绝对路径

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

#### 3.6.2.2 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="@{/show}">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>
```



### 3.6.3 在 url 中实现参数传递

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

### 3.6.4 在 url 中通过 restful 风格进行参数传递

```
<a th:href="@{/path/{id}/show(id=1,name=zhangsan)}"> 相 对 路 径 - 传 参  
-restful</a>
```