Thymeleaf

5. 设置属性值

属性	作用
th:text	计算其值表达式并将结果设置为标签的标签体
th:utext	th:text 会对结果中的特殊字符转义，th:utext 不会转义
th:attr	为标签中的任意属性设置，可以一次设置多个属性
th:*	为 html 指定的属性设值,一次设置一个
th:alt-title	同时为 alt 与 title 属性赋值
th:lang-xmllang	同时为 lang 、xml：lang 属性赋值
th:fragment	定义模板片段
th:insert	将被引用的模板片段插入到自己的标签体中
th:replace	将被引用的模板片段替换掉自己
th:include	类似于 th:insert，而不是插入片

	段，它只插入此片段的内容
th:remove	删除模板中的某些代码片段
th:each	迭代数据 ,如 数组、List、Map 等
th:if	条件为 true 时,显示模板片段，否则不显示
th:unless	条件为 false 时,显示模板片段，否则不显示
th:switch	与 Java 中的 switch 语句等效，有条件地显示匹配的内容
th:case	配合 th:switch 使用
th:with	定义局部变量
th:inline	禁用内联表达式，内联 js,内联 css

5.1 设置任何属性值 th:attr

th:attr 提供了更改标签属性值的能力。th:attr 使用比较少，因为他的使用比较难，语法不优雅。对于标签的特定属性，请使用 th:value, th:action, th:href,th:class,th:src,th:onclick 等等.

参考文档：

https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#setti

ng-attribute-values

th:abbr	th:accept	th:accept-charset
th:accesskey	th:action	th:align
th:alt	th:archive	th:audio
th:autocomplete	th:axis	th:background

还有很多.....

1)创建 AttrController

```
@Controller
public class AttrController {

    @GetMapping("/attr")
    public String attr(Model model){
        model.addAttribute("myaction","/user/login");
        model.addAttribute("mytext","请登录");
        return "/attr";
    }
}
```

2)创建 attr.html

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

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style type="text/css">
        .large-font{
            font-size: 20pt;
            color: red;
        }
    </style>
</head>
<body>
    <p>设置属性值</p>
    <form action="/hello.html" th:attr="action=${myaction}">
        账号：<input type="text" name="username"><br/>
        密码：<input type="text" name="pwd"><br/>
        <input type="submit" value="登录"
th:attr="value=${mytext}">
    </form>
    <br/>
    <img src="没有想好" th:attr="src=@{/image/im1.jpg},title='图
片',alt='可能不存在'" />
```

```
    <br/>
    <br/>
    <p>设置更多特定属性</p>
    <form th:action="@{/index.html}" th:method="get">
        <input type="text" th:name="uname"
th:value="zhangsan"><br/>
    </form>
    <br/>
    <a th:href="@{/hello.html}" th:class="large-font">访问
hello</a>
</body>
</html>
```

5.2 同时设置多个值

th:alt-title ：设置 alt ， title 两个属性

th:lang-xmllang：设置 lang ，xml:lang

例如 1：页面如下

```
<img th:src="@{/image/im1.jpg}" th:alt-title="我的图片" />
```

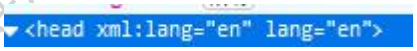
结果：

```
<img src="/image/im1.jpg" alt="我的图片" title="我的图片">
```

例如 2：页面如下

```
<head th:lang-xmllang="en">
```

结果：



5.3 boolean 属性

HTML 具有布尔属性的概念，例如 readonly，还有 checkbox 的“checked”，这个属性不赋值，没有值的属性意味着该值为“true”。也可以使用属性名本身表示 true， 即 checked=”checked”.

1）AttrControlelr 增加

```
model.addAttribute("selected",true);

model.addAttribute("unselect",false);
```

2）attr.html 增加

```
<!--${selected}为 true，设置 checked 为 checked

    ${unselect}为 false，不设置 checked 属性

-->

<input type="checkbox" value="游泳" th:checked="${selected}">游泳

<input type="checkbox" value="骑行" th:checked="${unselect}">骑行
```

5.4 设置标签体文本

th:text , th:utext.

th:text：用来计算表达式并将结果设置标签体，会对计算结果中特殊字符进行转义。

th:utext: 用来计算表达式并将结果设置标签体,不转义

1）创建 BodyController

```
@GetMapping("/body/text")
public String text(Model model){
        model.addAttribute("msg","学习的开发语言<b>java</b>等等");
        model.addAttribute("content","全栈开发
<b>vue</b>,<b>java</b>,<b>mysql</b>");
        return "/body";
}
```

2）创建 body.html

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

```
<body>
    <p th:text="${msg}">java,go 语言</p>
    <p th:utext="${content}">java 全栈</p>
</body>
</html>
```

5.5 循环 th:each

th:each 处理循环，类似 jstl 中的<c:foreach>。

特点：

①：循环的对象如果是 null，不存在则不循环。

②：循环包含自身和标签内全部内容。

③：可以遍历的对象：

数组；任何实现 java.util.Iterable 接口的对象；Enumeration 枚举；实现 Map 接口对象

语法格式：

```
<tr th:each="成员遍历:${表达式}">
    <td th:text="${成员}"> 列 </td>
</tr>
```

使用 th:each 时，Thymeleaf 提供了一种用于跟踪迭代状态的机制：状态变量。状态变量在每个 th:each 属性中定义，并包含以下数

据：

①index 属性：当前迭代索引，从 0 开始

②count 属性：当前的迭代计数，从 1 开始

③size 属性：迭代变量中元素的总量

④current 属性：每次迭代的 iter 变量，即当前遍历到的元素

⑥even/odd 布尔属性：当前的迭代是偶数还是奇数。

⑦first 布尔属性：当前的迭代是否是第一个迭代

⑧last 布尔属性：当前的迭代是否是最后一个迭代。

1) 创建数据类 Student

```
public class Student {
    private Integer id;
    private String name;
    private Integer age;
    private String className;

    public Student() {
    }

    public Student(Integer id, String name, Integer age, String
className) {
        this.id = id;
```

```
        this.name = name;
        this.age = age;
        this.className = className;
    }
}
```

2) 在 BodyController 增加

```
@GetMapping("/body/each")
public String eachList(Model model){
    List<Student> students = new ArrayList<>();
    Student s1 = new Student(1001,"黄忠",20,"一班");
    Student s2 = new Student(1002,"魏延",20,"二班");
    Student s3 = new Student(1003,"刘备",20,"三班");
    Student s4 = new Student(1004,"关于",20,"四班");
    Student s5 = new Student(1005,"张飞",20,"五班");
    students.add(s1);
    students.add(s2);
    students.add(s3);
    students.add(s4);
    students.add(s5);
    model.addAttribute("stulist",students);
```

```
//map
Map<String,String> map=new HashMap<>();
map.put("id","学号");
map.put("name","学生姓名");
map.put("age","年龄");
map.put("classname","所在班级");
model.addAttribute("stumap",map);

//list-map
List<Map<String,Student>> listmap = new ArrayList<>();
Map<String,Student> map1 = new HashMap<>();
map1.put("stu-1-1",s1);
map1.put("stu-2-2",s2);
listmap.add(map1);

Map<String,Student>    map2 = new HashMap<>();
map2.put("stu-2-1",s3);
map2.put("stu-2-2",s4);
map2.put("stu-2-3",s5);
listmap.add(map2);
```

```
        model.addAttribute("listmap",listmap);


        //普通数组
        String names[]= new String[] {"曹操","刘备","孙权"};
        model.addAttribute("names",names);



        //select 组件添加值
        Map<String,String> city = new HashMap<>();
        city.put("010","北京");
        city.put("021","上海");
        city.put("022","天津");
        city.put("023","重庆");
        model.addAttribute("citys",city);
        model.addAttribute("choice","北京");
        return "/each";
}
```

3）each.html 视图

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

```
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body>
   <p>each 循环 List</p>
   <table>
        <thead>
          <tr>
              <td>学号</td>
              <td>姓名</td>
              <td>年龄</td>
              <td>班级</td>
          </tr>
        </thead>
        <tbody>
          <tr th:each="stu:${stulist}">
              <td th:text="${stu.id}">学号</td>
              <td th:text="${stu.name}">姓名</td>
              <td th:text="${stu.age}">年龄</td>
              <td th:text="${stu.className}">班级</td>
```

```
            </tr>
        </tbody>
    </table>
    <p>each 循环 map</p>
    <div>
        <p th:each="m:${stumap}">
            <span th:text="${m.key}"></span>
            <span th:text="${m.value}"></span><br/>
        </p>
    </div>

    <p>each 循环 List<Map></p></body>
    <div>
        <ul th:each="lm:${listmap}">
            <li th:each="entry:${lm}" th:text="${entry.key}"></li>
            <li th:each="entry:${lm}"
th:text="${entry.value}"></li>
        </ul>
    </div>
    <p>each 循环 String[]</p>
    <di>
```

```
        <ul >
            <li th:each="name:${names}" th:text="${name}"></li>
        </ul>
    </di>

    <p>select 组件</p>
    <select>
        <option value="0">请选择</option>
        <option th:each="cm:${citys}" th:value="${cm.key}"
th:text="${cm.value}"
                th:selected="${cm.value} eq ${choice}" >城市
</option>
    </select>

    <p>使用状态变量</p>
    <table>
        <tr th:each="stu, loopStatus:${stulist}">
            <td
th:text="${loopStatus.count}+'/'+${loopStatus.size}"></td>
            <td th:text="${stu.id}">学号</td>
            <td th:text="${stu.name}">姓名</td>
```

```
            <td th:text="${stu.age}">年龄</td>
            <td th:text="${stu.className}">班级</td>
            <td th:text="${loopStatus.odd}?'奇数行':'偶数行
'"></td>
            <td th:text="${loopStatus.odd}"></td>
        </tr>
    </table>
</body>
</html>
```

5.6.判断 th:if 和 th:unless

th:if 当条件满足时，显示代码片段。 条件常用 boolean 表示，true 满足，反之不满足。

thymeleaf 中，true 不是唯一满足条件。

1）如果表达式结果为布尔值，则为 true 或者 false

2）如果表达式的值为 null，th:if 将判定此表达式为 false

3）如果值是数字，为 0 时，判断为 false；不为零时，判定为 true

4）如果值是 String，值为 “false”、“off”、“no” 时，判定为 false，否则判断为 true，字符串为空时，也判断为 true

5）如果值不是布尔值，数字，字符或字符串的其它对象，只要不为 null，则判断为 true

th:unless 是不满足条件显示代码片段， 类似 java 中 if 的 else 部分。

1.在 BodyController 增加方法

```
@GetMapping("/body/if")
public String testIf(Model model){

    //true
    model.addAttribute("old",true);
    model.addAttribute("login","login");
    model.addAttribute("str0","");
    model.addAttribute("num1",10);
    model.addAttribute("num2",-2);
    model.addAttribute("obj1",new Student());
    //false
    model.addAttribute("yong",false);
    model.addAttribute("str1","off");
    model.addAttribute("str2","no");
    model.addAttribute("str3","false");
    model.addAttribute("num2",0);
    model.addAttribute("obj2",null);
    return "/if";
}
```

2）创建 if.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body>
    <p th:if="true">boolean 类型  th:if=true</p>
    <p th:if="5>0">th:if=5>0</p>
    <p th:if="'ok'">字符串 ok</p>
    <p th:if="99">99 数字</p>
    <p th:if="'true'">'true'字符串</p>
    <p th:if="${old}">old 是 true</p>
    <p th:if="${login}">登录了</p>
    <p th:if="${num1}">num1=10</p>
    <p th:if="${num2}">num2=-2</p>
    <p th:if="${obj1}">obj1 is not null</p>
    <hr/>
    <p th:unless="${yong}">yong false</p>
    <p th:unless="${str1}">str1=off</p>
    <p th:unless="${str2}">str1=no</p>
```

```
    <p th:unless="${str3}">str1=false</p>
    <p th:unless="${num2}">num2=0</p>
    <p th:unless="${obj2}">obj2 is null</p>
</body>
</html>
```

5.7 th:switch 和 th:case

th:switch 就是 java 中 switch 语句，th:case 是匹配条件，只有一个执行，当多个 th:case 都为 true，只取第一个。

th:case=”*” 等同于 default。

1）BodyController 增加方法

```
@GetMapping("/body/switch")
public String testSwitch(Model model){
    model.addAttribute("marray",true);
    model.addAttribute("state",1);//0 初始订单， 1， 已成功，2，已经收货。
    return "/switch";
}
```

2）创建 switch.html

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

```
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body>
    <div th:switch="${marray}"> 结婚没有
        <p th:case="true">已婚</p>
        <p th:case="false">未婚</p>
        <p th:case="*">未知</p>
    </div>
    <br/>
    <br/>
    <div>
        <ul th:switch="${state}"> 订单状态：
            <li th:case="0">初始订单</li>
            <li th:case="1">已成功</li>
            <li th:case="2">已经收货</li>
            <li th:case="*">无信息</li>
        </ul>
    </div>
```

```
</body>
</html>
```

5.7 模板使用

模板就是公用资源，可以多次重复使用的内容。 经常把页眉，页脚，菜单做成模板，在各个其他页面使用。

模板使用，先定义再使用。可以在当前页面定义模板，也可在其他页面中定义模板。

1) 定义模板语法：

```
<div th:fragment="模板名称">
    模板内容
</div>
```

2) 引用模板

① 把模板插入到当前位置

```
<div insert="模板所在文件名称::模板名称">
    其他内容
</div>
```

②把模板替换当前标签内容

```
<div replace="模板所在文件名称::模板名称">
```

其他内容

③模板内容添加到当前标签

其他内容

3）模板引用语法

①：模板所在文件名称::模板名称

②：~{模板所在文件名称::模板名称}

4）模板作为函数形式使用

```
<div th:fragment="funtpl(one,two)">
    <p th:text="'hello' +   ${one} +'-' + ${two}"> </p>
</div>
```

th:insert ,th:replace 使用 funtpl 模板，可以传入值。

```
<div th:replace="frag/footer::funtpl(one='张三',two='李四')">
    我是参数模板
</div>
```

5）删除模板

th:remove=”删除范围值”

1）all：删除包含标签及其所有子项

2）body：不删除包含标签，但删除所有的子项

3）tag：删除包含标签，但不要删除其子项

4）all-but-first：删除第一个子项以外的其它所有子项

5)none :什么都不做。该值对于动态计算有用。null 也会被视为 none

模板使用例子：

1. 在 templates 目录下创建子目录 frag

在 frag 目录下创建 footer.html，内容如下：

```
<!DOCTYPE html>
<html lang="en" xmlns:th="www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div th:fragment="header">
    <b>欢迎大家，蛙课网全栈工程师</b>
</div>
```

```
<div th:fragment="copy">
    &copy; 蛙课网 2018-2020 北京动力节点
</div>


<div th:fragment="funtpl(one,two)">
    <p th:text="'hello' +    ${one} +'--' + ${two}">啊啊</p>
</div>
</body>
```

在 frag 目录下创建 main.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head th:lang-xmllang="en">
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>我是 main.html</p>


    <div th:insert="frag/footer::${fragname}">头部模板</div>
```

```
    <br/>
    <br/>
    <br/>
    <p>定义片段模板</p>
    <p th:fragment="course">这是 Thymeleaf 学习课程</p>
    <p id="tplId">我们学习的是模板</p>
    <br/>
    <br/>
    <div th:insert="frag/footer :: copy"></div>
    <div th:insert="~{frag/footer :: copy}"></div>
    <br/>
    <br/>
    <br/>
    <div th:insert="::course"></div>
    <div th:insert="~{::course}"></div>
    <br/>
    <br/>
    <div th:insert="::#tplId"></div>
    <br/>
    <br/>
    <div th:replace="::course">我是 replace 模板</div>
```

```
<br/>
<br/>
<div th:include="frag/footer::copy">
    我是 include
</div>
<br/>
<br/>
<p>参数模板</p>
<div th:replace="frag/footer::funtpl(one='张三',two='李四')">
    我是参数模板
</div>
<br/>
<br/>
<!--删除全部 本身和子标签-->
<div th:remove="all">
    指定 all
    <p>hello
        <span>world</span>
    </p>
</div>
```

```
<ul th:remove="body" >
    指定 body
    <p> welcome
        <span>thymleaf</span>
    </p>
</ul>

<div th:remove="tag">
    指定 tag
    <p>   hello
        <span>tag</span>
    </p>
</div>

<div th:remove="all-but-first">
    指定 all-but-first
    <p> 第一个子标签    </p>
    <p> 第二个子标签    </p>
</div>

<div th:remove="none">
```

```
        指定 none
        <p> 第一个子标签    </p>
        <p> 第二个子标签    </p>
    </div>
</body>
</html>
```

5.8 th:inline 使用

需要在 thymleaf 表达式写到标签体中，而不是标签内，可以使用

1）内联语法。

[[...]] 或 [(...)] 内联表达式，任何在 th:text 或 th:utext 属性中使用的表达式都可以出现在 [[]] 或 [()] 中使用，。

[[...]] 等价于 th:text； [(...)] 等价于 th:utext

2）禁用内联 原样输出的内容

3) 使用 javascript 内联

```
<script type=”text/javascript th:inline=”javascript”>
        js 代码 ， 内联表达式
</script>
```

例：

1）在 BodyContoller 增加方法

```
@GetMapping("/body/inline")
public String inline(Model model) {
    List<Student> students = new ArrayList<>();
    Student s1 = new Student(1001, "黄忠", 20, "一班");
    Student s2 = new Student(1002, "魏延", 20, "二班");
    Student s3 = new Student(1003, "刘备", 20, "三班");
    Student s4 = new Student(1004, "关于", 20, "四班");
    Student s5 = new Student(1005, "张飞", 20, "五班");
    students.add(s1);
    students.add(s2);
    students.add(s3);
    students.add(s4);
    students.add(s5);
    model.addAttribute("students",students);

    model.addAttribute("name","张三丰");
    model.addAttribute("info","我爱你<b>中国</b>！！！");
    return "/inline";
}
```

2）新建 inline.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
    <script type="text/javascript" th:inline="javascript">
        var jsname=[[${name}]];
        console.log("jsname:"+jsname)
        var stulist =[[${students}]];
        for(var i=0;i<stulist.length;i++){
            console.log("i="+i+" ," +stulist[i].name);
        }
    </script>
</head>
<body>
    <!-- th:text [[...]]   , th:utext=[(...)]-->
    <p th:text="${name}">我是常用写法</p>
    <p>内联语法：[[${name}]]</p>
    <br/>
    <p th:utext="${info}">我是常用写法</p>
    <p>内联语法：[(${info})]</p>
```

```
    <br/>
    <p>禁用内联</p>
    <p th:inline="none">[[我就是要这样显示]]</p>
    <p th:inline="none">[(我是要这样显示)]</p>

</body>
</html>
```

5.9 th:with 局部变量

th:with=”变量名 1=值 1,变量名 2=值 2” ，定义的变量只在当前标签内有效。

1）BodyController 增加方法

```
@GetMapping("/body/var")
public String useVar(Model model){
    model.addAttribute("age",20);
    return "/var";
}
```

2）创建 var.html

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

```
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body>
    <div th:with="name='zhangsan'">定义变量</div>
    <p th:text="${name}">使用变量</p>
    <br/>
    <!--html 注释-->

    <!--/*
    <div th:with="id=1002,name='lisi',myage=${age}">
        <p th:text="${id}"></p>
        <p th:text="${name}"></p>
        <p th:text="${myage}"></p>
    </div>
    */-->
</body>
</html>
```

5.9 工具类对象

官网地址：

https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-b-expression-utility-objects

1）#{execInfo}：模板信息

${#execInfo.templateName} 模板名称

${#execInfo.templateMode} 模板的处理模式

2）#uris

处理 url/uri 编码解码

${#uris.escapePath(uri)} //编码码

${#uris.escapePath(uri, encoding)} //指定编码转码

${#uris.unescapePath(uri)} //解码

${#uris.unescapePath(uri, encoding)} //指定编码解码

3）#dates：java.util.Date 对象的实用程序方法，可以操作数组，set，list。常用方法

${#dates.format(date, 'dd/MMM/yyyy HH:mm')}

${#dates.arrayFormat(datesArray, 'dd/MMM/yyyy HH:mm')}

${#dates.listFormat(datesList, 'dd/MMM/yyyy HH:mm')}

${#dates.setFormat(datesSet, 'dd/MMM/yyyy HH:mm')}

4）#numbers：数字对象的实用程序方法

${#numbers.formatInteger(num,size)} num 表示被格式的数字 ,size 表示整数位最少保留几位

5）#strings String 工具类，就是字符串工具类

{#strings.toUpperCase(name)} 大写

${#strings.toLowerCase(name)} 小写

${#strings.arrayJoin(namesArray,',') 连接，合并

${#strings.arraySplit(namesStr,',')} 分隔

${#strings.indexOf(name,frag)} 查找

${#strings.substring(name,3,5)} 取子串

${#strings.contains(name,'ez')} 是否有子串

${#strings.isEmpty(name)} 空判断

6)其他的还有 **#ids，#aggregates ,#maps ,#sets，#lists ，#arrays ， #bools ，#objects**

例子：

1）在 BodyController 增加方法

```
@GetMapping("/body/toolsobject")
public String toolsObject(Model model) {
```

```
        model.addAttribute("mydate",new Date());
        model.addAttribute("price",89.35);
        model.addAttribute("myname","zhangsan");
        return "/toolsobject";
    }
```

2）创建 toolsobject.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body
th:with="myurl='http://localhost:8080/myweb/hello.jsp?name=张三
'">
    模板名称：<p th:text="${#execInfo.templateName}"></p>
    模板模式：<p th:text="${#execInfo.templateMode}"></p>
    <br/>
    <div th:with="espath=${#uris.escapePath(myurl)}" >
        <p>编码：[[${espath}]]</p>
```

```
        <p>解码：[[${#uris.unescapePath(espath)}]]</p>
    </div>

    <br/>


    原始日期：  <p th:text="${mydate}"></p>
    <p th:text="${#dates.format(mydate,'yyyy-MM-dd HH:mm:ss')}"/>
    <br/>
    原始价格：<p th:text="${price}"></p>
    <p th:text="'1:' + ${#numbers.formatInteger(price,0)}"/>
    <p th:text="'2:' +${#numbers.formatInteger(price,3)}"/>
    <p th:text="'3:' +${#numbers.formatInteger(price,4)}"/>
    <p th:text="'4:' +${#numbers.formatDecimal(price,2,1)}"/>
    <br/>
    <p>字符串：[[${myname}]]</p>
    <p th:if="${#strings.isEmpty(myname)}">myname 不是null</p>
    <p th:text="${#strings.substring(myname,0,2)}"></p>
    <p th:text="${#strings.indexOf(myname,'san')}"></p>
```

```
    <p th:text="${#strings.toUpperCase(myname)}"></p>


</body>
</html>
```

5.10 内部对象

1）在 BodyContorller 增加方法

```
@GetMapping("/body/inobject")
public String inObject(Model model, HttpSession session){
        model.addAttribute("attrName","我是 attrValue");
        session.setAttribute("sessionAttr","我是 session 中的值");
        session.getServletContext().setAttribute("contextAttr","我是
context 中的值");
        return "/inobjects";

}
```

2）创建 inobjects.html

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

```
    <meta charset="UTF-8">
    <title>文本处理</title>
</head>
<body>
    <p>#request:请求对象，访问
javax.servlet.http.HttpServletRequest 对象</p>
    <p>contextPath: [[${#request.getContextPath()}]]</p>
    <p th:text="${#request.getParameter('name')}">参数 name 的值
</p>
    <p th:text="${#request.getAttribute('attrName')}">request 作用
域</p>
    <br/>
    <br/>
    <p>#session : 直接访问与当前请求关联的
javax.servlet.http.HttpSession 对象</p>
    <p th:text="${#session.getAttribute('sessionAttr')}">session 作用
域</p>
    <p th:text="${#session.getId()}">id</p>
    <br/>
    <p>#servletContext : 直接访问与当前请求关联的
javax.servlet.ServletContext 对象</p>
```

```
    <p
th:text="${#servletContext.getAttribute('contextAttr')}"></p>
    <p th:text="${#servletContext.getServerInfo()}"></p>
    <br/>
    <p>hymeleaf 在 web 环境中，有一系列的快捷方式用于访问请求参数、会话属性等应用属性
        param, request,session,访问它们而无需#
    </p>
    <br/>
    <p>param：用于检索请求参数。 ${param.foo}是一个使用 foo 请求参数的值 String[] ，所以${param.foo[0]} 将会 通常用于获取第一个值。</p>
    <p>param 获取参数： [[${param.foo[0]}]]</p>
    <p>param 获取参数： [[${param.foo[1]}]]</p>
    <p>参数数量：[[${param.size()}]]</p>
    <p>是否有指定参数:[[${param.containsKey('name')}]]</p>
    <p>参数值： [[${param.userid}]]</p>
    <br/>
    <p>session:用于获取 session 属性。与 param 同理，只是作用域不同而已</p>
    <p>作用域值的: [[${session.sessionAttr}]]</p>
```

```
    <p>session 中有几个值：[[${session.size()}]]</p>
    <br/>
    <p>application：用于获取应用程序或 servlet 上下文属性。与
param 同理。</p>
    <p>ServletContext 作用域值的:
[[${application.contextAttr}]]</p>
    <p>application 中有几个值：[[${application.size()}]]</p>
</body>
</html>
```

访问的 url 地址：

http://localhost:8080/body/inobject?name=zhangsan&userid=1234&foo=11&foo=aaaa