Session 27

ThymeLeaf

Reading and Reference

- Reading
  www.baeldung.com/thymeleaf-in-spring-mvc

- Reference:
  - API
    https://www.thymeleaf.org/apidocs/thymeleaf/2.1.6.RELEASE/index.html
Lecture Objectives

- Light covering of topics in ThymeLeaf
  - Expressions
  - Form handling
  - Text replacement

Thymeleaf has much more material than can be covered in one or two sessions.

Recap

- We used Thymeleaf in the Spring Model session to display a message stored in the Spring model object.
- We refer to `${message}` as an expression that accesses the value of a model attribute.

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  ...
  <body>
    <h1 th:text="${message}"></h1>
  </body>
</html>
```

Recall that we accessed the Spring model in the Spring controller.
**EL Variables**

${product}$

- The web container evaluates a variable that appears in an expression by looking up its value.
- For example, when evaluating the expression ${product}$, the container will look for the name “product” in the page, request, session, and application scopes and will return its value (in the first scope in which it encounters the value).
- If “product” is not found, null is returned.
- A variable that matches one of the implicit objects will return that implicit object instead of the variable’s value.

**EL Syntax**

```
Today is: ${b.d.hours}
```

- **EL Implicit Object**
  - pageScope
  - requestScope
  - sessionScope
  - applicationScope
  - param
  - paramValues
  - header
  - headerValues
  - cookie
  - initParam
  - pageContext

- **Attribute name**
  - In pageScope
  - In requestScope
  - In sessionScope
  - In applicationScope

Either a map key or a bean property (depending on whether b is a Map or a bean)

A variable on the left side of a dot is either a Map (something with keys) or a bean (something with properties)

This works if an object exhibits “bean-like” behavior

b.d evaluates to either a Map value or a bean property value (d is either a Map key or a bean property name)
th:text Example

To illustrate that the Spring model can be set at run time

```java
@RequestMapping(value="/thmodel", method=RequestMethod.GET)
String getThmessage(Model m2) {
    m2.addAttribute("serverTime", new Date());
    return "thsample5";
}
```

```
thsample5.html
...<h1 th:text="${serverTime}"></h1>...
```

th:text attribute evaluates its value expression and sets the result as the body of the host tag - replacing the existing body

Alternate Thymeleaf Syntax

- The following are equivalent
- The second line is referred to as the data attribute syntax
- Sometimes used since the second form is considered pure HTML5

```
<h1 th:text="${serverTime}"></h1>
```

```
<h1 data-th-text="${serverTime}"></h1>
```
Expression Syntax Conditional Operators

- Literals, text operations, arithmetic operations, Boolean operations, comparisons, equality operations

- Conditional operators:
  - If-then: `(if) ? (then)`
  - If-then-else: `(if) ? (then) : (else)`
  - Default: `(value) ?: (defaultValue)`

Expression Basic Objects

- `#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

<p>The locale is <span th:text="${#locale.country}"/></span>.</p>
<p>The request method is <span th:text="${#request.method}"/></span>.</p>

Notice syntax difference with EL

The locale is US.
The request method is GET.
Expressions Within th:text

- An alternate way of coding

  ```html
  <p>The request method is
  <span th:text="${#request.method}"/>
  </p>
  ```

  Is

  ```html
  <p th:text="'The request method is ' + ${#request.method} + '.'">
  </p>
  ```

---

Expression Asterisk Syntax

- Variable expressions can also be written as *{...}*

- This evaluates an expression of selected objects, instead of the whole context.
Collection Attributes

If the model attribute is a collection of objects, the th:each tag attribute can be used for iteration over the collection

```html
<ul>
<li th:each="hname: ${#request.headerNames}">
    <span th:text="${hname}"/>
</li>
</ul>
```

For each element of ${#request.headerNames}, repeat this fragment, using the current element in a variable called hname

Conditional Evaluation

Thymeleaf version of if statement

Used to display a section of the view if the condition is met

Corresponding unless element

```html
<p>
    <span th:each="hname: ${#request.headerNames}"
        th:if="${hname}=='host'" th:text="${hname}"
    >
        found host header
    </span>
</p>
```

Sample use of th:if

host
Handling User Input

- You can inject your form fields into a bean that is accessible in your Thymeleaf template.

```java
@Controller
@RequestMapping(value = "displaypage")
public class CardController {
    @RequestMapping(value = "/formexample")
    public String saveCard(
        @ModelAttribute Card card, Model model)
    {
        return "thsample6";
    }
}

class Card {
    private String cnum;
    private String nickname;
    // getters and setters
}
```

Asterisk Expression Syntax - Example

- Your template refers to your bean, and bean attributes are accessible.
- The result of an expression using the `th:object` attribute is a selected object.
- `*{}` expression syntax operates on selected objects.

```html
<form method="put" action="..." id="form1" th:object="${card}"
  
  <td>
    <input type="text" id="cnum" name="cnum" th:field="*{cnum}" />
  </td>
</form>
```
Are We on Track?

- Using your project, display the names and values of your form components in a 2-column html table. The table should have only as many rows needed to exactly display all form parameters.

Hints - Alternate ways to do this:

1. Consider using the methods of the HttpServletRequest expression basic objects (#request) (look at the API).
2. Look for Map and Map.Entry.
3. Consider injecting parameters into the model in your controller.

Thymeleaf Track

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnum</td>
<td>123</td>
</tr>
<tr>
<td>nickname</td>
<td>Allonzo</td>
</tr>
<tr>
<td>rds</td>
<td>Adult</td>
</tr>
<tr>
<td>submits</td>
<td>Submit Library Card Application</td>
</tr>
</tbody>
</table>

Were We on Track?

- Using a basic object

```html
<table>
  <thead>
    <tr><th>Property</th><th>Value</th></tr>
  </thead>
  <tbody>
    <tr th:each="aname: ${#request.parameterMap}"
      th:if="!#{request.getParameter(aname.key)}">
      <td>${aname.key}</td>
      <td><span th:text="${#request.getParameter(aname.key)}"></span></td>
    </tr>
  </tbody>
</table>
```

Map.Entry object

```html
<table>
  <tbody>
    <tr>
      <td><span th:text="${aname.key}"></span></td>
      <td><span th:text="${#request.getParameter(aname.key)}"></span></td>
    </tr>
  </tbody>
</table>
```