SECJ 3303 – INTERNET PROGRAMMING

TOPIC 5 – SPRING WEB MVC WITH THYMELEAF INTEGRATION

innovative • entrepreneurial • global

OBJECTIVES

Applied
- Creating Dynamic Web Pages Using Spring Web MVC and Thymeleaf

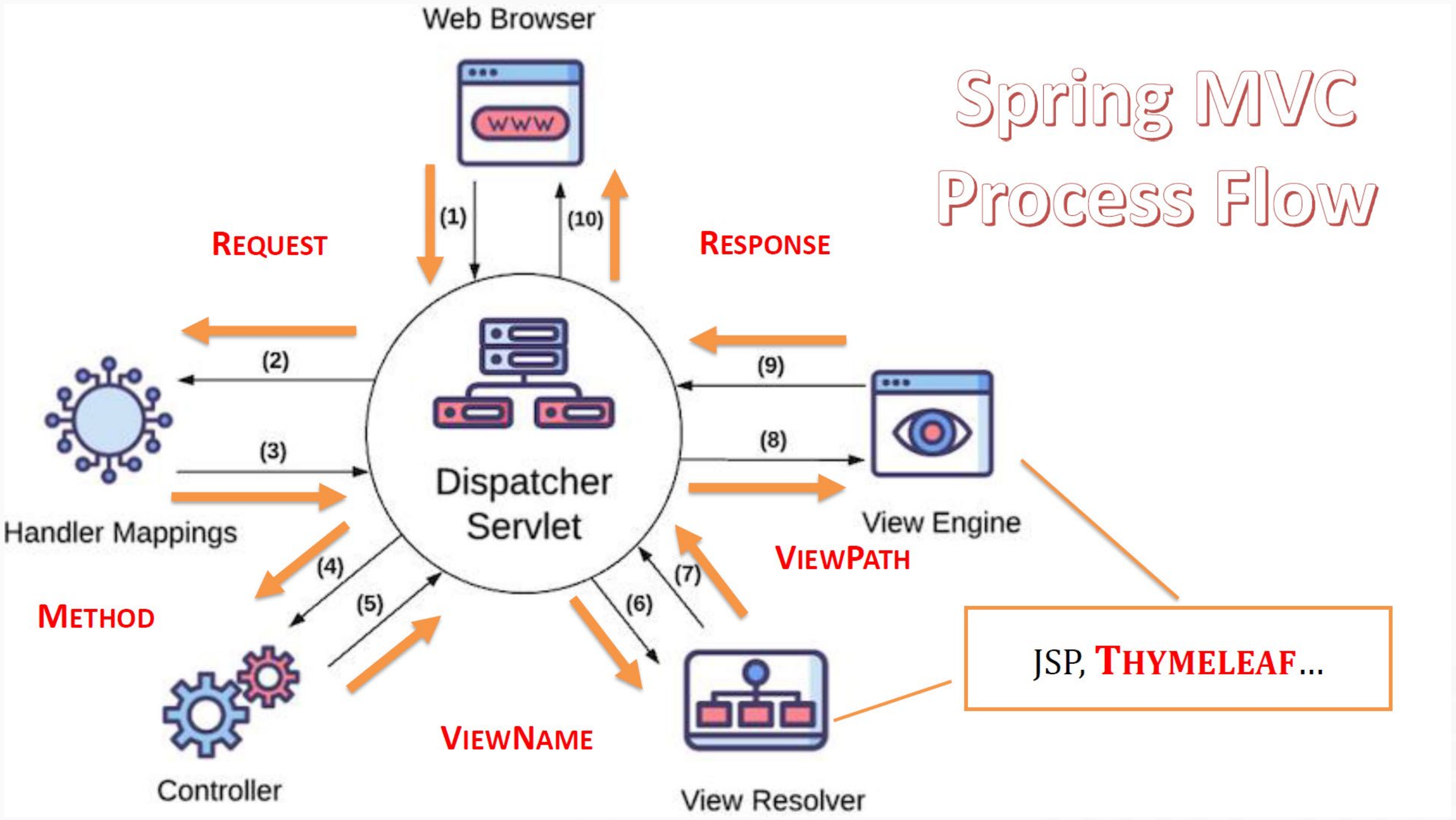
Knowledge
- Introduction to Thymeleaf as a View Layer for Spring Applications.
- Passing Data Between Controllers and Thymeleaf Views
- Configuring View Resolvers in Spring (Thymeleaf Integration)

Thymeleaf

- Thymeleaf is a modern server-side Java template engine for both web and standalone environment, capable of processing HTML, XML, JavaScript, CSS and even plain text.
- It is designed for generating web content dynamically.
- It's commonly used to generate HTML views for web application.
- The main goal of Thymeleaf is to provide an elegant and highly-maintainable way of creating templates.
- Benefits:
  - Easy to use, manage, and maintain.
  - Improves collaboration between design and development teams.

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

Thymeleaf


Web Browser
Spring MVC Process Flow
(1)
(10)
REQUEST
RESPONSE
(2)
(3)
(9)
(8)
Dispatcher Servlet
Handler Mappings
View Engine
VIEWPATH
(4)
(5)
(6)
(7)
METHOD
JSP, THYMELEAF...
Controller
VIEWNAME
View Resolver

JSP vs Thymeleaf

JSP

Suitable for web pages but less flexible for frontend developers.

Thymeleaf

Offers flexibility for both web and non-web environments with .html files.

JSP

Pros	Cons
Web page development	Difficult for frontend developers
Dynamic content	Specific file extension

Thymeleaf

Pros	Cons
Versatile usage	Learning curve
Non-web compatibility	Limited community support
HTML file format	

Setup and Configuration

Thymeleaf dependencies in pom.xml

```
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
<dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>5.3.19</version>
</dependency>

<!-- Thymeleaf Dependency -->
<dependency>
            <groupId>org.thymeleaf</groupId>
            <artifactId>thymeleaf</artifactId>
            <version>3.1.1.RELEASE</version>
</dependency>

<!-- Thymeleaf Spring Integration -->
<dependency>
            <groupId>org.thymeleaf</groupId>
            <artifactId>thymeleaf-spring5</artifactId>
            <version>3.1.1.RELEASE</version>
</dependency>
<!-- Servlet API Dependency -->
<dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
</dependency>
```

Setup and Configuration

Spring Configuration file = [servletname]-servlet.xml

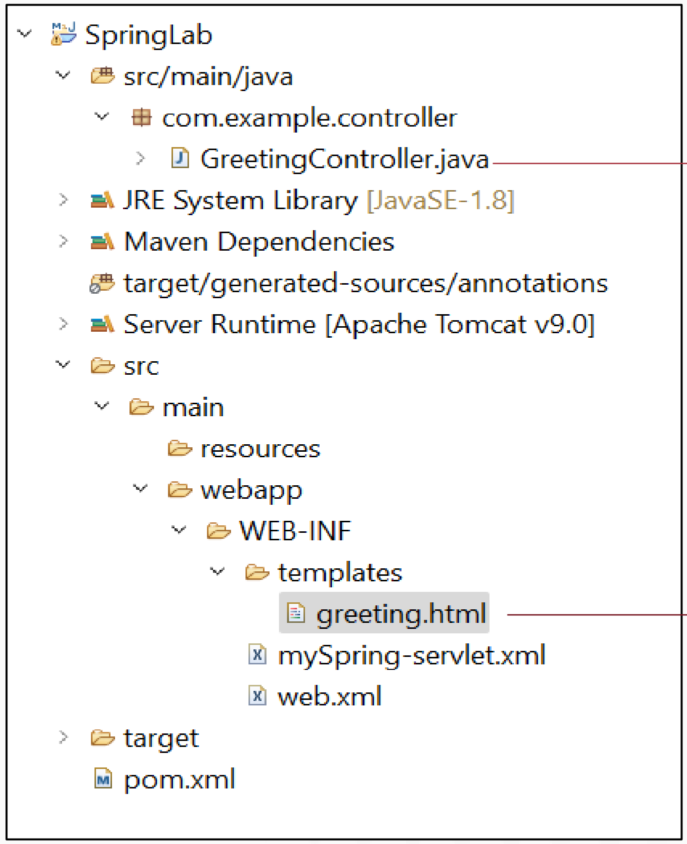
```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation=" http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/mvc
           http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<mvc:annotation-driven/>
<!-- Scan for Controllers -->
<context:component-scan base-package="com.example.controller"/>

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

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

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

</beans>
```

Setup and Configuration

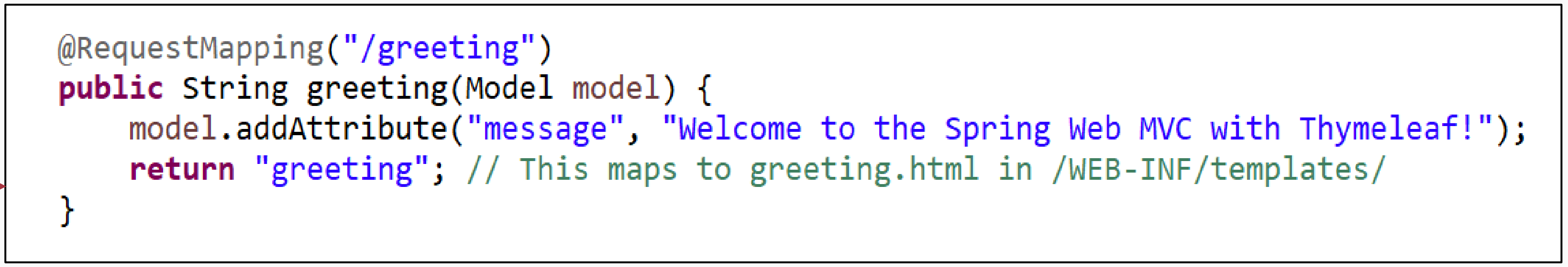
Dispatcher Servlet= web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">

<!-- DispatcherServlet Configuration -->
<servlet>
  <servlet-name>mySpring</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>


<!-- URL Pattern Mapping -->
<servlet-mapping>
   <servlet-name>mySpring</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>

</web-app>
```

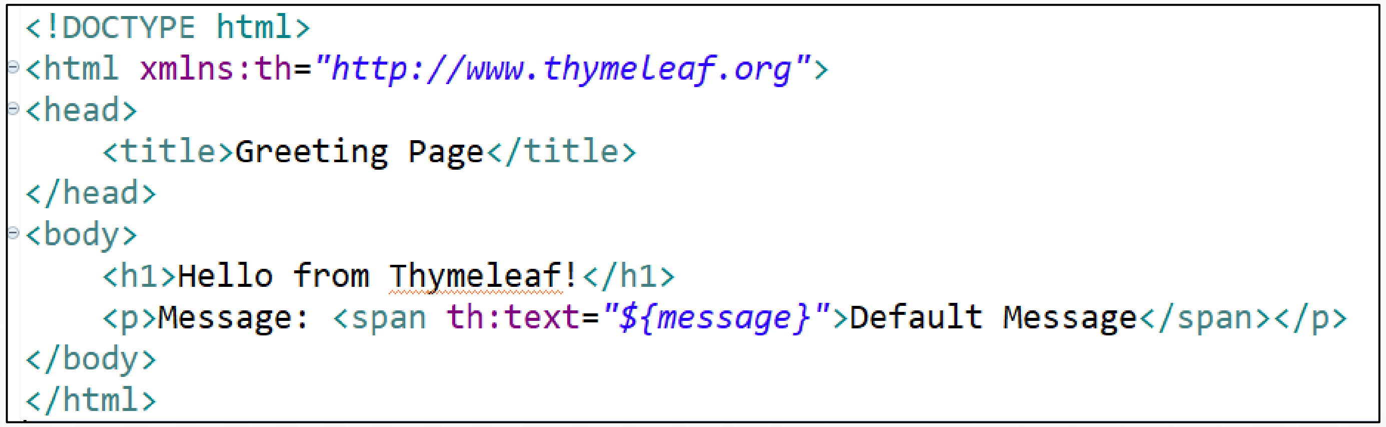
Setup and Configuration



```
SpringLab
  src/main/java
    com.example.controller
      GreetingController.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  target/generated-sources/annotations
  Server Runtime [Apache Tomcat v9.0]
  src
    main
      resources
      webapp
        WEB-INF
          templates
            greeting.html
          mySpring-servlet.xml
          web.xml
  target
  pom.xml
```



```java
@RequestMapping("/greeting")
public String greeting(Model model) {
    model.addAttribute("message", "Welcome to the Spring Web MVC with Thymeleaf!");
    return "greeting"; // This maps to greeting.html in /WEB-INF/templates/
}
```



```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Greeting Page</title>
</head>
<body>
    <h1>Hello from Thymeleaf!</h1>
    <p>Message: <span th:text="${message}">Default Message</span></p>
</body>
</html>
```

Thymeleaf
Structure Directory

Thymeleaf Page Structure and Syntax

Defining Namespace:

Use <html xmlns:th="http://www.thymeleaf.org"> in HTML.

Example of Syntax Usage:

- **Text Rendering:**
- <span th:text="“ ${message} "></span>
- <span>[[*${message}*]]</span>
- *Generated HTML: <span>Faculty&lt;b&gt;Computing&lt;/b&gt;</span>*

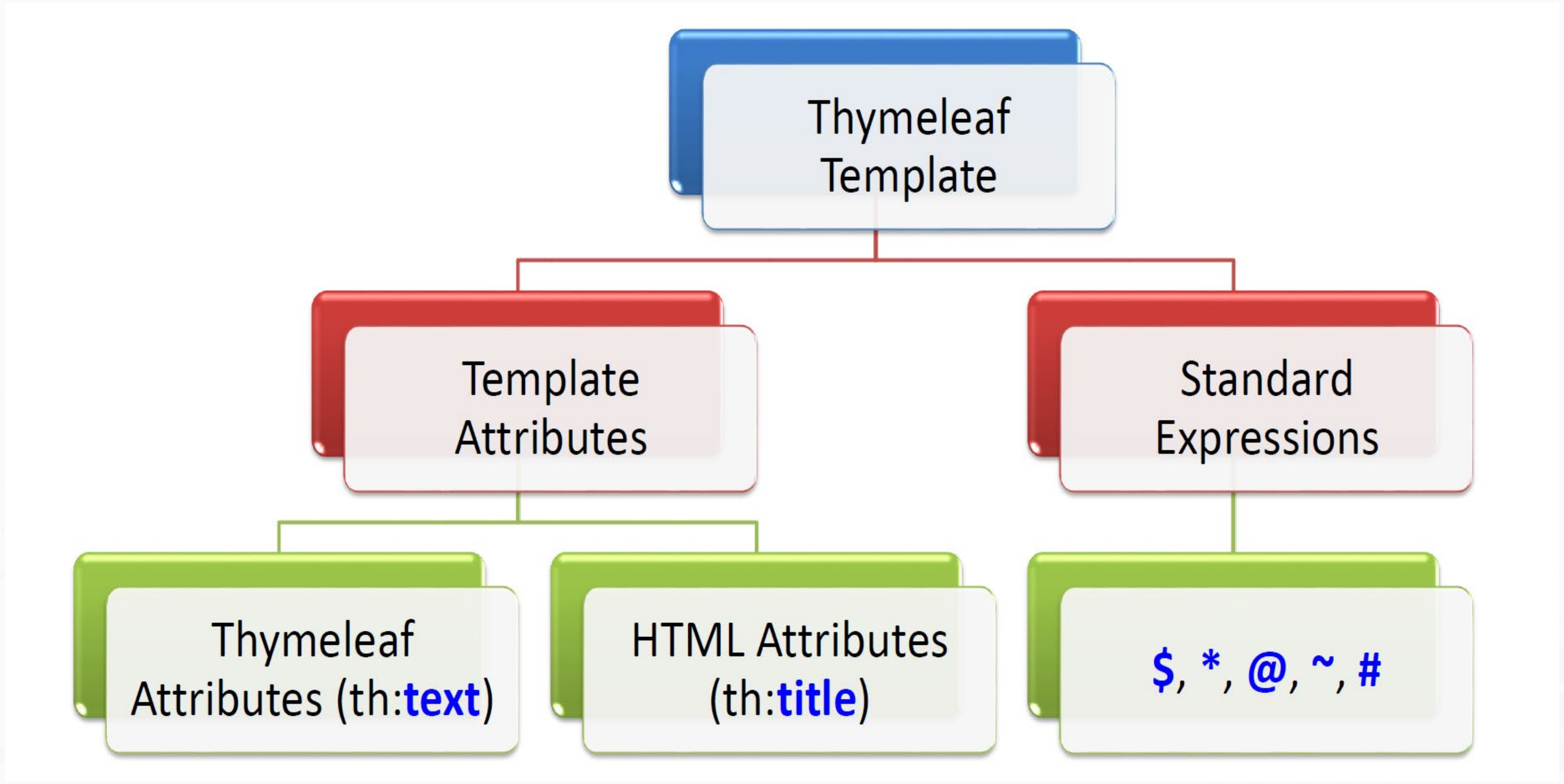
Faculty <b>Computing</b>

- **Unescaped Text:**
- <span th:utext="“ ${message} "></span>
- <span>[[*${message}*]]</span>
- *Generated HTML: <span>Faculty <b>Computing</b></span>*

Faculty **Computing**

Thymeleaf Template

th attribute = “Standard Expression”



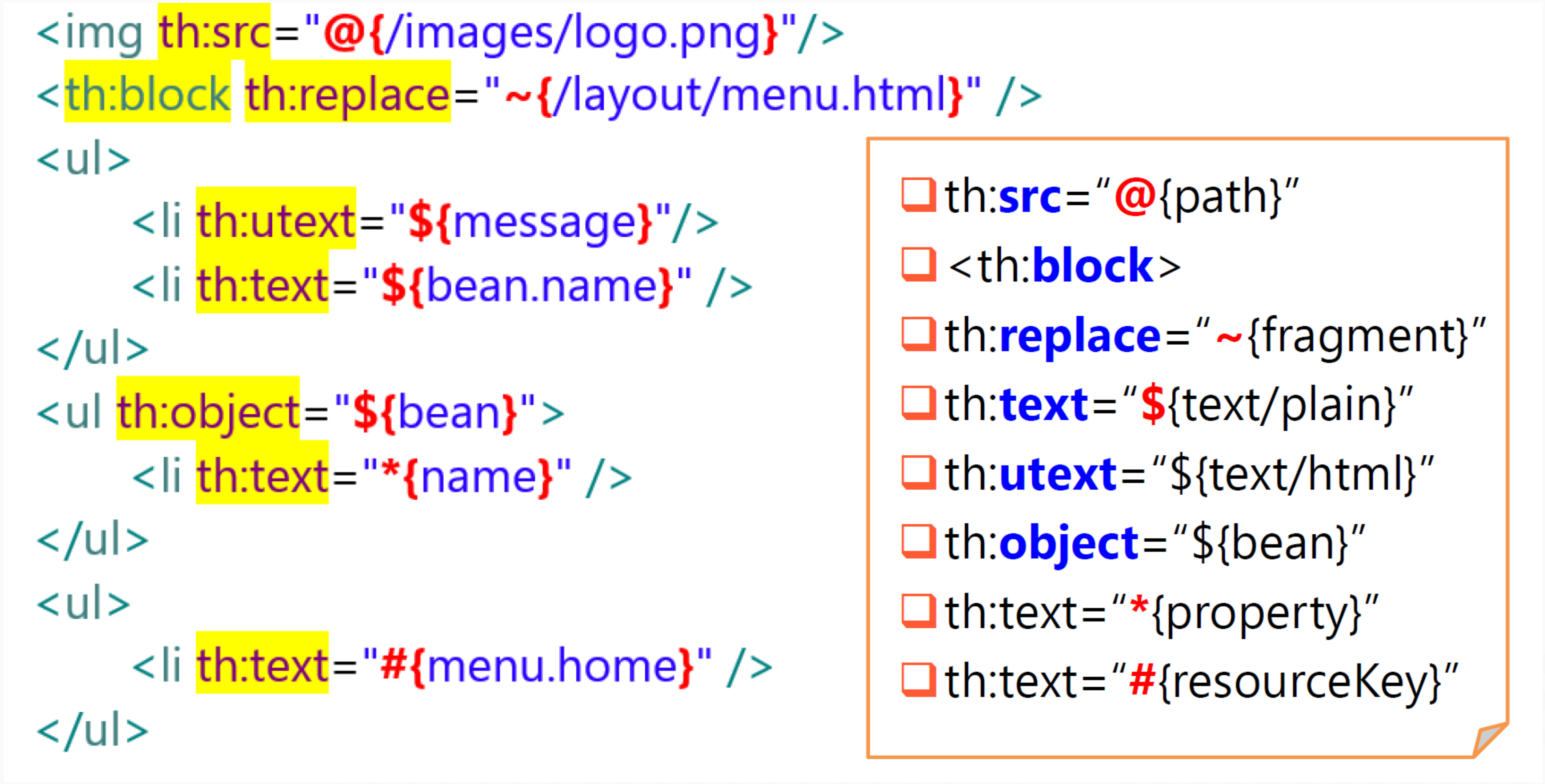
Thymeleaf Attributes

Action	Attributes
Include fragments	th:insert, th:replace
Loop	th:each
Conditional evaluation	th:if, th:unless, th:switch, th:case
Define local variables	th:object, th:with
Modify attributes	th:attr, th:attrprepend, th:attrappend
Modify HTML attributes	th:value, th:href, th:src, etc.
Modify element content	th:text, th:utext
Declare fragments	th:fragment
Remove fragments	th:remove

Thymeleaf Standard Expressions

- ❑ Variable Expressions (${...}):
  - ❑ Retrieve variables from the model (e.g., ${user.name})
- ❑ Selection Expressions (*{...}):
  - ❑ Work with properties of the current bound object in th:object.
- ❑ Message Expressions (#{...}):
  - ❑ Access localized messages from resource files (e.g., #{menu.home}).
- ❑ URL Expressions (@{...}):
  - ❑ Generate URLs dynamically based on the context or parameters.
- ❑ Fragment Expressions (~{...}):
  - ❑ Embed reusable content or fragments in templates

Standard Expressions Example



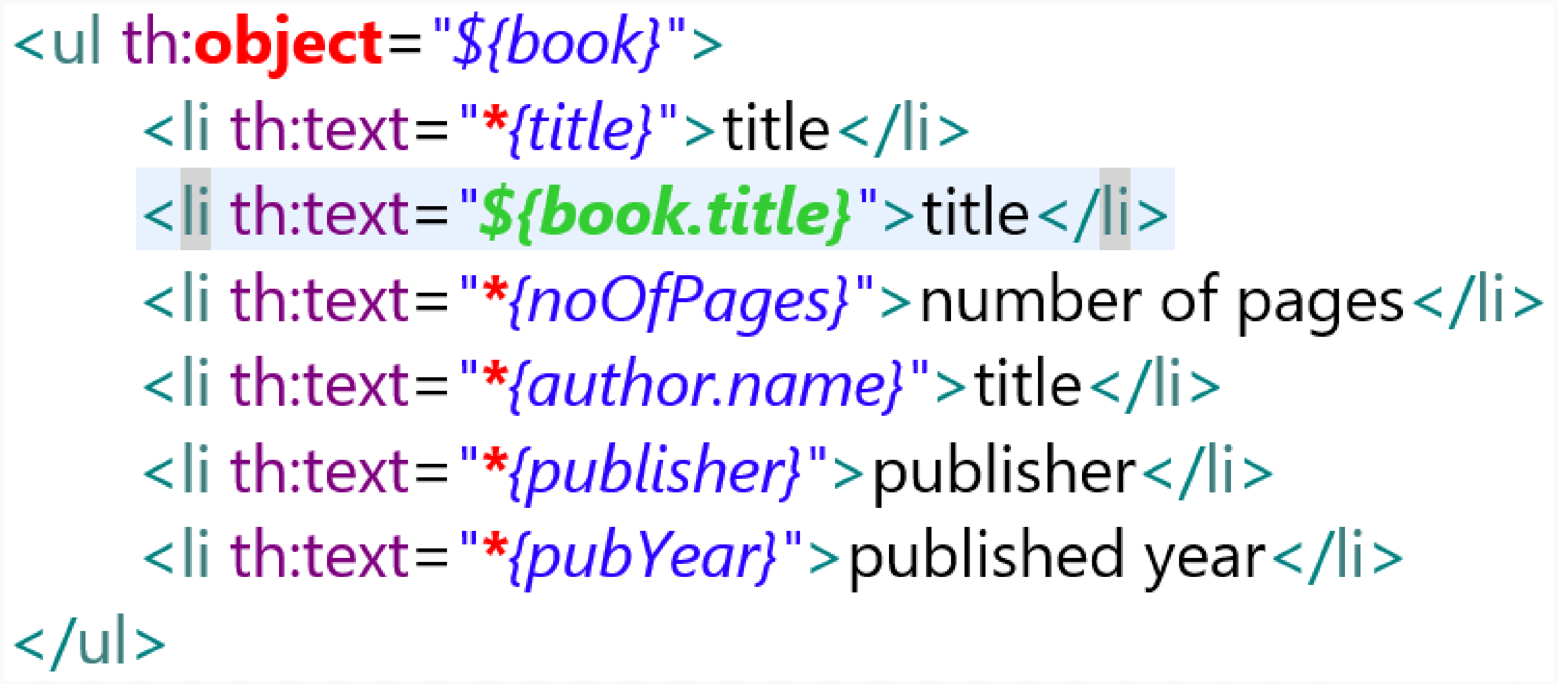
```html
<img th:src="@{/images/logo.png}"/>
<th:block th:replace="~{/layout/menu.html}" />
<ul>
    <li th:utext="${message}"/>
    <li th:text="${bean.name}" />
</ul>
<ul th:object="${bean}">
    <li th:text="*{name}" />
</ul>
<ul>
    <li th:text="#{menu.home}" />
</ul>
```

- th:src=“@{path}”
- <th:block>
- th:replace=“~{fragment}”
- th:text=“${text/plain}”
- th:utext=“${text/html}”
- th:object=“${bean}”
- th:text=“*{property}”
- th:text=“#{resourceKey}”

Standard Expressions : 1-Variable Expressions ${...}

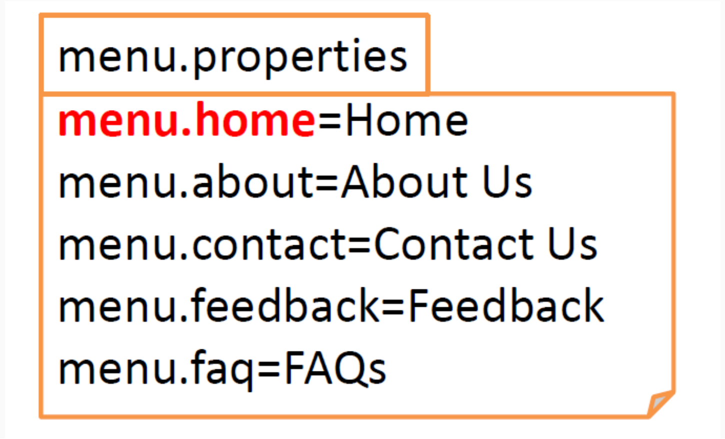
```
<ul>
        <li th:text="${message}" th:title="${message}"></li>
        <li th:text="${session.message}"></li>
        <li th:text="${application.message}"></li>
</ul>
<ul>
        <li th:text="${bean.name}"/>
        <li th:text="${bean.salary}"/>
        <li th:text="${bean.gender}"/>
</ul>
<ul>
        <li th:text="${param.name}"/>
</ul>
```

Standard Expressions : 2-Selection Expressions *{...}

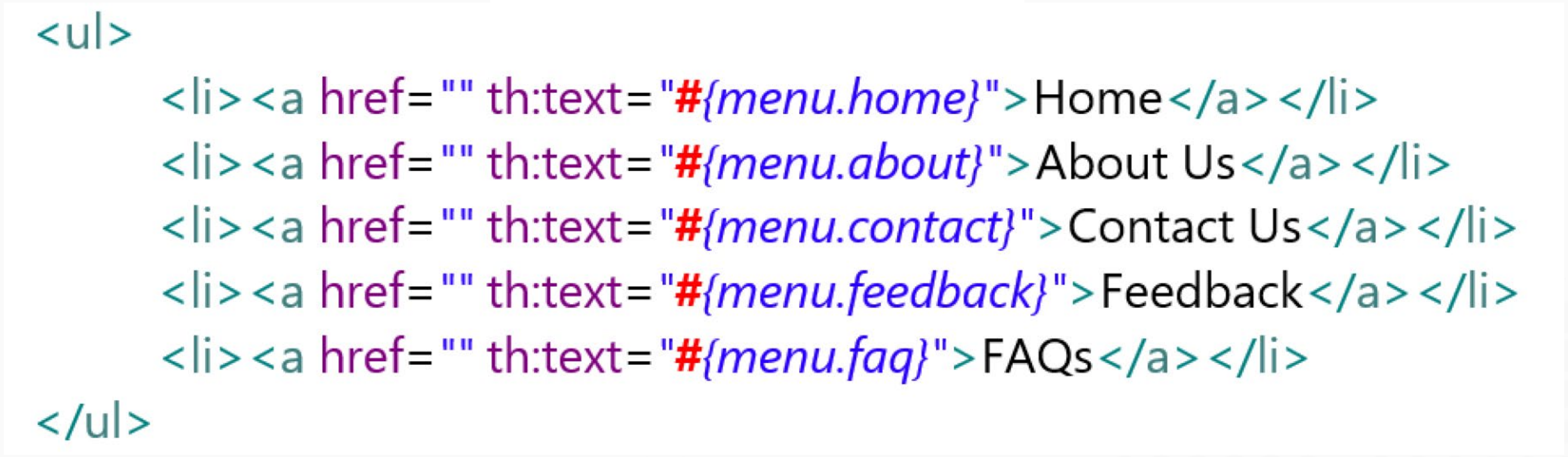


```
<ul th:object="${book}">
     <li th:text="*{title}">title</li>
     <li th:text="${book.title}">title</li>
     <li th:text="*{noOfPages}">number of pages</li>
     <li th:text="*{author.name}">title</li>
     <li th:text="*{publisher}">publisher</li>
     <li th:text="*{pubYear}">published year</li>
</ul>
```

Standard Expressions : 3-Message Expressions #{...}



```
menu.properties
menu.home=Home
menu.about=About Us
menu.contact=Contact Us
menu.feedback=Feedback
menu.faq=FAQs
```



```
<ul>
    <li><a href="" th:text="#{menu.home}">Home</a></li>
    <li><a href="" th:text="#{menu.about}">About Us</a></li>
    <li><a href="" th:text="#{menu.contact}">Contact Us</a></li>
    <li><a href="" th:text="#{menu.feedback}">Feedback</a></li>
    <li><a href="" th:text="#{menu.faq}">FAQs</a></li>
</ul>
```

Standard Expressions :
4-Link (URL) Expressions @{...}

- ❑ **Root Relative URL** (relative to the Webroot)

```
<a th:href="@{/order/list}">...</a>
➞ <a href="/ctxpath/order/list">...</a>
```

- ❑ **Page Relative URL** (relative to the current URL)

```
<a th:href="@{../order/list}">...</a>
➞ <a href="../order/list">...</a>

<a th:href="@{order/list}">...</a>
➞ <a href="order/list">...</a>
```

- ❑ **Protocol Relative and Absolute URL**

```
<a th:href="@{//www.utm.my/order/list}">...</a>
➞ <a href="//www.utm.my/order/list">...</a>

<a th:href="@{https://www.utm.my/order/list}">...</a>
➞ <a href="https://www.utm.my/order/list">...</a>
```

Parameters and Path Variables

❑ Parameters

```
<th:block th:with="x='X', y='Y'">
    <a th:href="@{/order/details(a=${x},b=${y})}">...</a>
    <a th:href="@{|/order/details?a=${x}&b=${y}|}">...</a>
    <a th:href="@{'/order/details?a=' + ${x} + '&b=' + ${y}}">...</a>
</th:block>
```

```
<a href="/order/details?a=X&b=Y">...</a>
```

❑ PathVariables

```
<th:block th:with="x='X', y='Y'">
    <a th:href="@{/order/{a}/details/{b}(a=${x},b=${y})}">...</a>
    <a th:href="@{|/order/${x}/details/${y}|}">...</a>
    <a th:href="@{'/order/' + ${x} + '/details/' + ${y}}">...</a>
</th:block>
```

```
<a href="/order/X/details/Y">...</a>
```

Standard Expressions : 5-Fragment Expressions ~{...}

Fragment expressions are used to copy a file or a predefined fragment template into desired locations.

```
<div th:insert="~{/menu.html}">...</div>
➡ Replaces the content of the div tag with the content of the file menu.html.

<div th:replace="~{/menu.html}">...</div>
➡ Replaces the entire div tag with the content of the file menu.html.

<div th:insert="~{/fragments.html :: menu}">...</div>
➡ Replaces the content of the div tag with the fragment named menu in the file
fragments.html.

<div th:replace="~{/fragments.html :: menu}">...</div>
➡ Replaces the entire div tag with the fragment named menu in the file
fragments.html
```

Thymeleaf Utility Object

- **Dates:** Helps format and display date values.
  Example: `${#dates.format(date, 'dd/MM/yyyy’)}`.

- **Numbers:** Format numerical data with precision.
  Example: `${#numbers.formatDecimal(1234.567, 2)}`.

- **Strings:** Provides string manipulation methods, like capitalization.
  Example: `${#strings.capitalize('hello world’)}`.

- **Lists/Arrays:** Offers operations for collections, such as finding size or length.
  Example: `${#lists.size(list)}`

Utility Object – Example

```
<ul th:object="${student}">
        <li>Fullname:
                <b th:text="*{#strings.capitalizeWords(fullname)}"></b></li>
        <li>Marks:
                <b th:text="*{#numbers.formatDecimal(marks, 0, 'COMMA', 2, 'POINT')}"></b></li>
        <li>Birthday:
                <b th:text="*{#dates.format(dob, 'dd-MM-yyyy')}"></b></li>
        <li th:if="*{marks >= 9.0}">Grade: <b>Golden Bee</b></li>
</ul>
```

```
@Data
public class Student {
        String fullname =   Ahmad
        Double marks = 9.5;
        Date dob = new Date();
}
```

- Fullname: **Ahmad**
- Marks: **9.50**
- Birthday: **23-02-2021**
- Grade: **Golden Bee**

Forms in Thymeleaf

- In Thymeleaf you can create almost normal HTML forms:

```
<form th:action="@{/user}" th:method="post">
       <input type="number" name="id"/>
       <input type="text" name="name"/>
       <input type="submit"/>
</form>
```

- You can have a controller that will accept an object of given type:

```
@PostMapping("/user")
public ModelAndView register(@ModelAttribute User user)
{ ... }
```

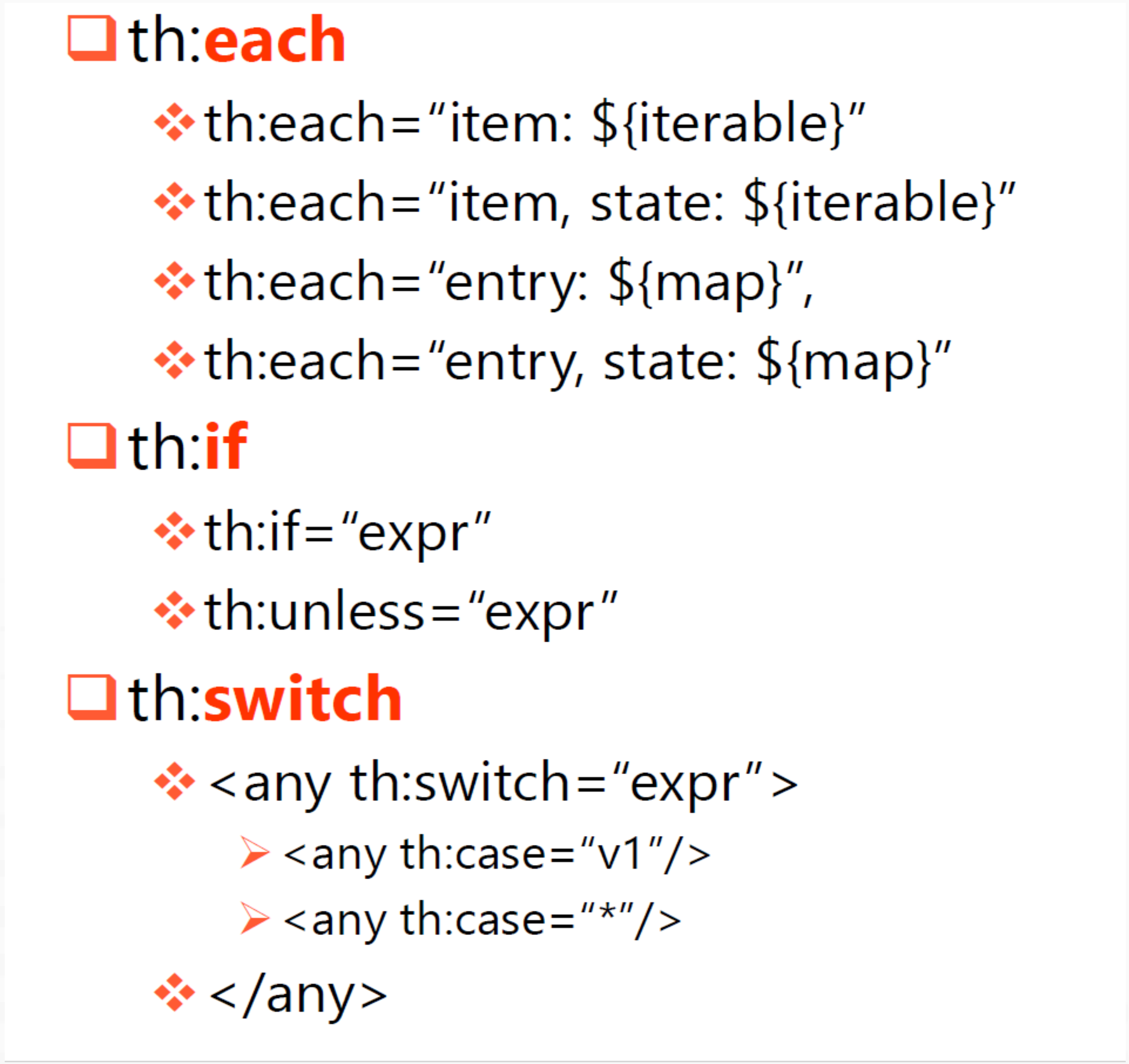
Forms in Thymeleaf

- You can pass objects to forms in order to use validations:

```
<form th:action="@{/user}" th:method="post" th:object=${user}>
        <input type="number" th:field="*{id}"/>
        <input type="text" th:field="*{name}"/>
        <input type="submit"/>
</form>
```

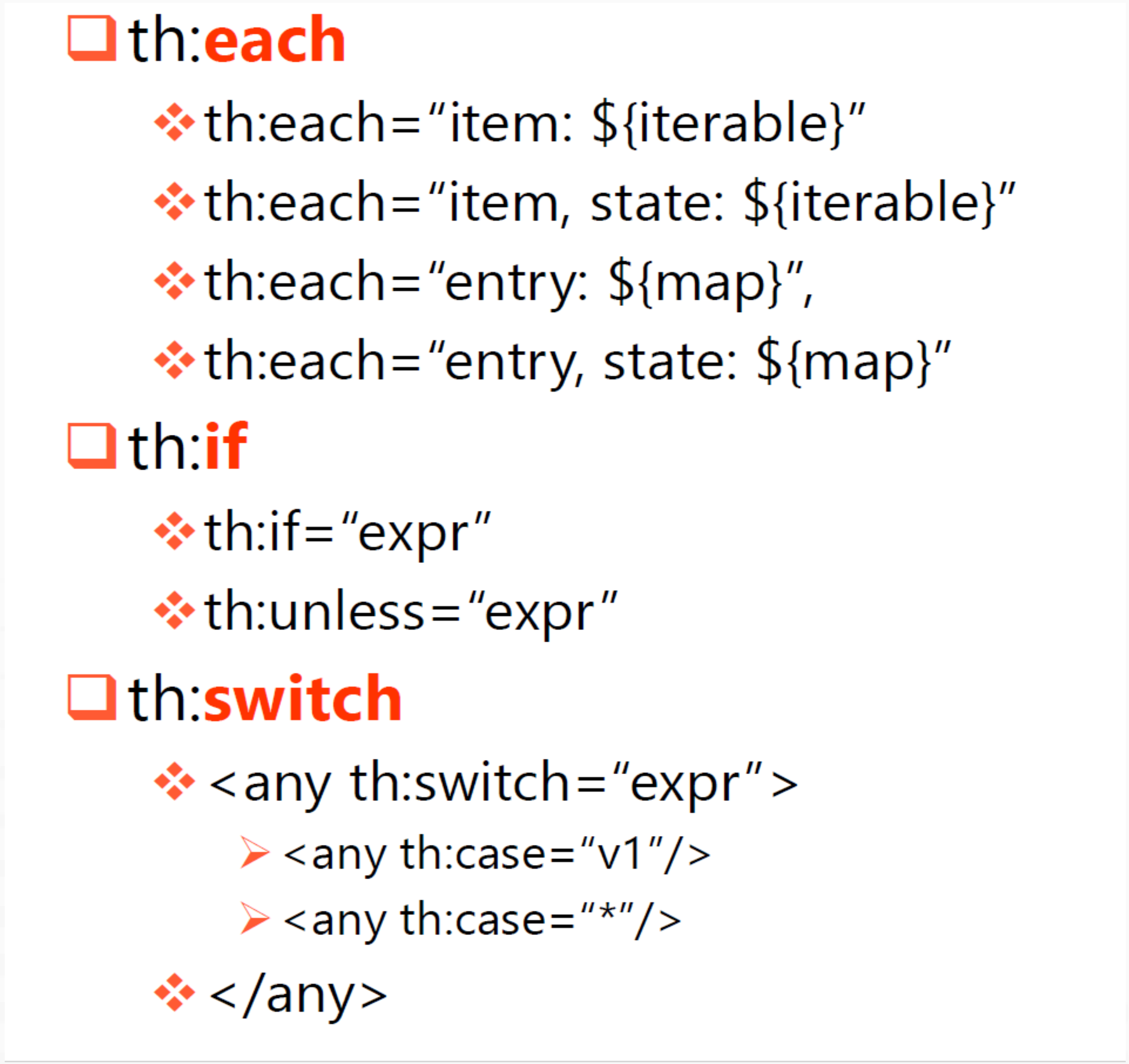
- The th:field attribute creates different attributes based on the input type.

Thymeleaf Flow Control Attributes



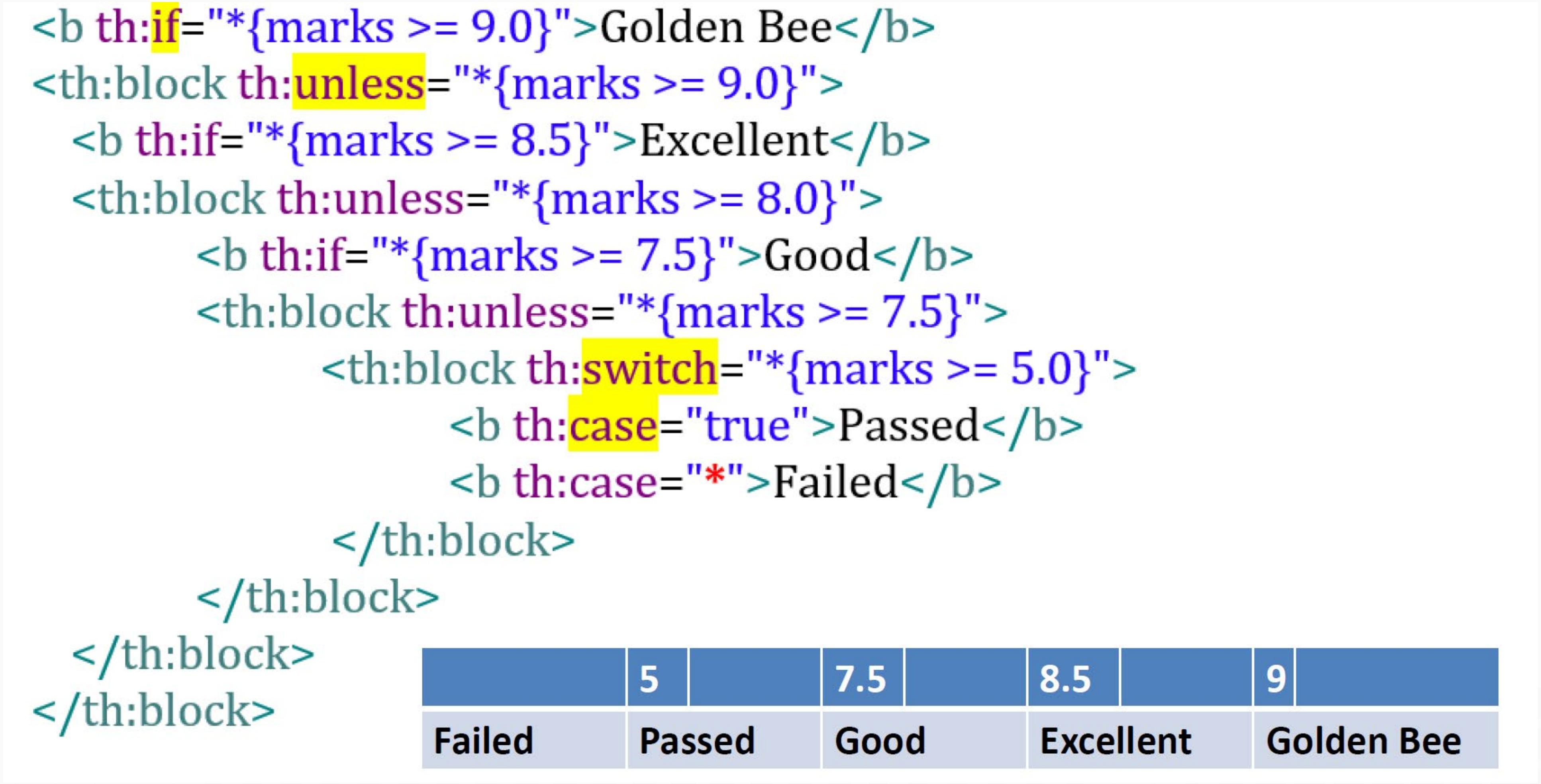
- th:**each**
  - th:each="item: ${iterable}"
  - th:each="item, state: ${iterable}"
  - th:each="entry: ${map}",
  - th:each="entry, state: ${map}"
- th:**if**
  - th:if="expr"
  - th:unless="expr"
- th:**switch**
  - <any th:switch="expr">
    - <any th:case="v1"/>
    - <any th:case="*"/>
  - </any>

Thymeleaf Flow Control Attributes



- th:**each**
  - th:each="item: ${iterable}"
  - th:each="item, state: ${iterable}"
  - th:each="entry: ${map}",
  - th:each="entry, state: ${map}"
- th:**if**
  - th:if="expr"
  - th:unless="expr"
- th:**switch**
  - <any th:switch="expr">
    - <any th:case="v1"/>
    - <any th:case="*"/>
  - </any>

Flow Control Attributes – Example



```
<b th:if="*{marks >= 9.0}">Golden Bee</b>
<th:block th:unless="*{marks >= 9.0}">
  <b th:if="*{marks >= 8.5}">Excellent</b>
  <th:block th:unless="*{marks >= 8.0}">
        <b th:if="*{marks >= 7.5}">Good</b>
        <th:block th:unless="*{marks >= 7.5}">
              <th:block th:switch="*{marks >= 5.0}">
                    <b th:case="true">Passed</b>
                    <b th:case="*">Failed</b>
               </th:block>
        </th:block>
  </th:block>
</th:block>
```

	5	7.5	8.5	9
Failed	Passed	Good	Excellent	Golden Bee

Flow Control Attributes - Example

```
<ul th:each="student: ${list}" th:object="${student}">
        <li>Fullname: <b th:text="*{fullname}"></b></li>
        <li>Marks: <b th:text="*{marks}"></b></li>
        <li>Grade:
                <b th:if="*{marks >= 9.0}">Golden Bee</b>
                <th:block th:unless="*{marks >= 9.0}">
                  <b th:if="*{marks >= 8.5}">Excellent</b>
                  <th:block th:unless="*{marks >= 8.0}">
                    <b th:if="*{marks >= 7.5}">Good</b>
                    <b th:unless="*{marks >= 7.5}">
                      [[*{marks >= 5.0 ? 'Passed' : 'Failed'}]]
                    </b>
                  </th:block>
                </th:block>
        </li>
</ul>
```

- Fullname: Maryam
- Marks: 9.5
- Grade: Golden Bee

- Fullname: Abu
- Marks: 8.5
- Grade: Excellent

- Fullname: Ahmad
- Marks: 7.5
- Grade: Good

- Fullname: Amin
- Marks: 5.0
- Grade: Passed

- Fullname: Ali
- Marks: 4.5
- Grade: Failed

Summary

- ❑ Overview of Thymeleaf template structure and syntax
- ❑ Setup and Configuration
- ❑ Use of namespaces, attributes, and expressions
- ❑ Standard expressions: ${}, *{}, @{}, ~{}, #{}
- ❑ Utility objects for dates, numbers, strings and lists/arrays
- ❑ Forms in Thymeleaf
- ❑ Flow Control attributes

TOPIC 5 – Spring Web MVC With Thymeleaf Integration

The End

innovative • entrepreneurial • global

darjat seseor
KERANA TUHAN UNTUK MANUSIA • UNIVERSITI TEKNOLOGI MALAYSIA •
UTM JOHOR BAHRU