Thymeleaf

==========

(As alternate to jsp pages)

=> Another UI Technology that is build on the top of html for dynamic web pages

=> only html gives static web pages.. html tags+ thymeleaf tags give dynamic webpages. It is light weight alternate to heavy weight jsp pages..

=> In the execution of jsp page lot of memory and lot of cpu time is required becoz interanlly translate's ajsp equalent servlet comp and creates multiple implicit objs (9) if used or not used.. for all these things lot of memory and cpu time required.

=>To overcome the above problems of jsp pages use thymeleaf as lightweight alternate for rendering dynamic webpages...

=> We need to write thymeleaf tags in html tags.. by importing thymeleaf namespace by specifying its namespace uri..

<html

</html>

mvc

xmlns:th="https://www.thymeleaf.org">

=>Spring boot gives built-in support of thymeleaf UI by giving default prefix is <classpath>/templates/

=>

default suffix is .html

(location)

(file extension)

=>html pages are static web pages =>html +thymeleaf pages are dynamic webpages => jsp pages are dynamic web pages note:: Thymeleaf namespace is the library that contains set of tags .every name space is identified with its namespace uri

=>Thymeleaf can be used only in java env.. =>thymeleaf file extension must be .html file

(classpath here is src/main/resources folder)

To use thymeleaf in spring boot add this starter to pom.xml

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf -->
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

<version>

</dependency>

-</version>

Current spring boot version

note:: we can not mixup thymeleaf and jsp UI togather in a single spring MVC or spring Boot MVC web application.

=>To execute thymeleaf code Thymeleaf engine is required

which will come becoz this jar file.. This engine converts thymeleaf tags to html code.. and sends to browser as response..

=> Standard jsp tags identified with fixed prefix called <jsp: xxxx> and similary thymeleaf tags are indentified with <th: xxx> prefix (In html tags the attributes will be prefixed with "th" to add thymeleaf support)

=> Popular symbols in thymeleaf programming

@ --->to specify Location (/<globalpath>/<request path>)

---> To read data from container managed scopes like model attributes *---> To bind/link data to form comps (useful only in thymeleaf forms) (model properties, ref data)

a. To read data from model attributes/container managed scopes

=> block scope, method scope, instance scope, class scope, thread scope are JVM managed scopes

=> request, session, application, page and etc.. are WebContainer managed scopes

=> singleton, prototype, request, session, application, websocket and etc.. are IOC container managed scopes

-> for primitive/wrapper/String type model attrbutes th:text="${<attribute name>}"

(eg:)

→> for object type model attrbutes th:text="${<objectName>}" eg<span th:text="${emp}"/>)

-> for getting property values from Object type model attrbutes th:text="${<objectName>.<property name>}" (eg: <span th:text="${emp.ename}"/>) -> for looping through arrays/collection th:each="<counter variable>:${<collection/array type attribute>1"

b. To display images (To link image file to <img> tag)

<img th:src="@{/path}">

c. To Link/map with hyperlink

**<a th:href="@{/path}"> xxxx **

d.To Link/map css file

<link rel="stylesheet" th:href="@{/path}">

e. To Link/map java script fle

<script type="text/javascript" th:src="@{/path}"> </script>

f) Thymeleaf forms ar bidirencing)

(eg: <table>

<tr th:each="cust:${custList}">

acts like for loop

<td> <span th:text="${cust.cno}"/> </td> <td> <span th:text="${cust.cname}"/> </td> </tr> </table>)

while working with thymeleaf, we must take controller having class level global path using @RequestMapping("/...") annotation.

(Global path)

[Global path for controller class is mandatory]

forms i.e they support DataBinding(writing form data to model class obj) and

DataRendering (writing handler methods supplied model attributes/model class obj data to form comps)

=> To specify action url <form_th:action="@global path/request path}" > (alternate to <frm:form

action="..."/>)

=> For binding model class obj data/model attributes data to form comps (for form backing object operation)

->

<form th:object="${model attribute name/object name of model clas}"> (alternate to <frm:from modelAttribute="....">)

=> For binding model class obj data /model attributes data to form comps

<input type="text" th:field="*<model class property name/model attrobute name>}"> (alternate to <frm:input path="..."/>)

(ref data)

What is the difference b/w

attrs

th:text and th:field.

in thymeleaf

==========

========:

data

(for Data rendering)

(for, Data Binding activity)

th:text is given to read. from different scopes and to display them on browser.. like reading and leaf displaying model class obj data and model attrbutes data.. (can be used in thyme forms and non-forms env..)
th:field is given to bind model class obj property values /model attribute values(reference data) to form comps (can be used only in thymeleaf forms)

Converting MiniProject to Thymeleaf UI based Application

=========

step1) add thymeleaf starter to pom.xml file

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

step2) Provide global path to controller class

@Controller

@RequestMapping("/employee") //global path public class EmployeeController {

...

}

using

*** we can remove**

JSTL jar files from pom.xml

note:: while thymeleaf tags .. the global path

for controller is mandatory.. (to avoid clashes between request paths of different controller class)

step3) copy js,images folders of webapp/webcontent to "static" folder of src/main/resources folder

and WEB-INF/pages folder jsp files to template folder of "src/main/resources" folder, change .jsp extension to .html

#src/main/resources static

✓ images

add.jfif

delete.jfif

edit.jpg home.jfif

templates

edit_employee.html home.html show_emps.html application.properties

we can comment view resolver cfg in application.properties ##View Resolver

#spring.mvc.view.prefix=/WEB-INF/pages/

#spring.mvc.view.suffix=.jsp

refer ✓ MS BootMVCProj20-MiniProject-CURD-Thymeleaf [boot]

note:: we can delete images, js folders of src/main/webapp

folder and src/main/webapp/WEB-INF/pages folder

to

step4) modify "template" folder .html files code thymeleaf code from jsp code.

step5) Run the Application...

home.html

```
<html xmlns:th="http://www.thymeleaf.org">
<h1 style="color:red;text-align: center">Spring boot MVC - Mini Project</h1>
<h2 style="text-align:center"><a th:href="@{/employee/emp_report}">Generate Employee Report</a></h2>
<br>
<center> <a th:href="@{/employee/emp report}"><img_th:src="@{/images/report.png}" width="100" height="100"></a></center>
```

show_employee_report.html

```
<html xmlns:th="http://www.thymeleaf.org">
<div th:if="${!empsList.empty}">
<h1 style="color:red;text-align:center"> Employees Report </h1>
<table border="1" align="center" bgcolor="cyan">
<tr style="color: red"><th>empno </th><th> emp name</th><th> Job </th> <th>salary </th><th> deptno</th> <th> operations</th> </tr>
<tr th:each="emp:${empsList}" style="color: blue">
<td><span th:text="${emp.empno}"/></td>
<td><span th:text="${emp.ename}"/></td>
<td><span th:text="${emp.job}"/></td> <td><span_th:text="${emp.sal}"/></td>
```

```
<td><span th:text="${emp.deptno}"/></td>
<td><a th:href="@{/employee/emp_edit(no=${emp.empno})}"><img th:src="@{/images/edit.png}" width="30px" height="30px"></a>
<a th:href="@{/employee/emp_delete(no=${emp.empno})}" onclick="return confirm('Do u want to delete the employee')"><img th:src="@{/images/delete.png}" width="30px" height="30px"></a> </td> </tr> </table>
</div>
<div th:if="${empsList.empty}">
<h1 style="color:red;text-align:center"> Employees Not found </h1>
</div>
<center>
<h2 style="color:green;text-align:center" th:text="${resultMsg}"/>
<a th:href="@{/employee/emp_add}"><img th:src="@{/images/add.png}" width="40px" height="50px"> Add Employee</a>     <a_th:href="@{/employee/}"><img_th:src="@{/images/home.png}" width="40px" height="50px">Home</a>
</center>
```

update_employee.html

```
<html xmlns:th="http://www.thymeleaf.org">
<h1 style="color:red;text-align:center">Update Employee</h1>
<form th:action="@{/employee/emp_edit}" th:object="${emp}" method="POST">
<table align="center" bgcolor="cyan">
<tr>
<td> Employee Number:: </td>
<td> <input type="text" th:field="*{empno}" readonly="true"/> </td>
</tr>
<tr>
<td> Employee Name:: </td>
<td> <input type="text" th:field="*{ename}"/> </td>
</tr>
<tr>
<td> Employee Desg:: </td>
<td> <input type="text" th:field="*{job}"/> </td>
</tr>
<tr>
<td> Employee salary::</td>
<td> <input type="text" th:field="*{sal}"/> </td>
</tr>
```

MMVCBootProj16-MiniProject-CURDOperations-thymeleaf [boot]

> Deployment Descriptor: MVCBootProj16-MiniProject-CURDOperation >Spring Elements

> JAX-WS Web Services

#src/main/java

✓ com.nt

> BootMvcProj07MiniProjectCurd OperationsApplication.java

>ServletInitializer.java

com.nt.controller

> EmployeeOperations Controller.java

com.nt.model

> Employee.java

com.nt.repository

> EmployeeRepository.java

com.nt.service

> EmployeeMgmtServiceImpl.java

> EmployeeMgmtService.java

src/main/resources

✓ static

images

add.png

→ delete.png

edit.png

home.png

report.png

templates

home.html

register_employee.html

show_employee_report.html

update_employee.html

application.properties

> #src/test/java

> JRE System Library [JavaSE-17]

<tr>

>

Maven Dependencies

<td> Dept no </td>

>

Deployed Resources

<td> <input type="text" th:field="*{deptno}"/> </td> </tr>

> src

> target

WHELP.md

mvnw

<tr>

mvnw.cmd Mpom.xml

<td> <input type="submit" value="Update Employee"></td>

<td> <input type="reset" value="cancel"> </td>

</tr>

</table>

</form>

register_employee.html

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

<h1 style="color:red;text-align:center">Register Employee</h1>

<form th:action="@{/employee/emp_add}" th:object="${emp}" method="POST">

<table align="center" bgcolor="cyan">

<tr>

<td> Employee Name:: </td>

<td> <input type="text" th:field="*{ename}"/> </td>

</tr> <tr>

<td> Employee Desg:: </td>

<td> <input type="text" th:field="*{job}"/> </td>

</tr>

<tr>

<td> Employee salary::</td>

<td> <input type="text" th:field="*{sal}"/> </td>

</tr>

<tr>

<td> dept no </td>

<td> <input type="text" th:field="{deptno}"/> </td>*

</tr>

<tr>

<td> <input type="submit" value="submit"></td>

<td> <input type="reset" value="cancel"> </td>

</tr>

</table>

</form>

Boot strap (It is css++ and java script ++)

=> It is library that gives set of readymade CSS styles and java script libraries

=> These styles can be applied on the applications directly to format the results

step1)

Link boot strap css by collecting from internet

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"

integrity="sha384-ggOyROiXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

crossorigin="anonymous"/>

step2) Apply CSS style classes at various levels

*** ***

<table class="table table-hover">

step3) Test the application