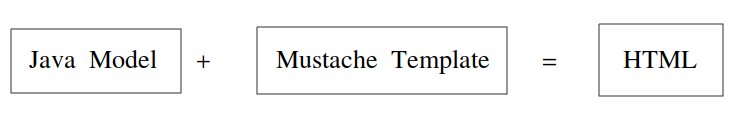
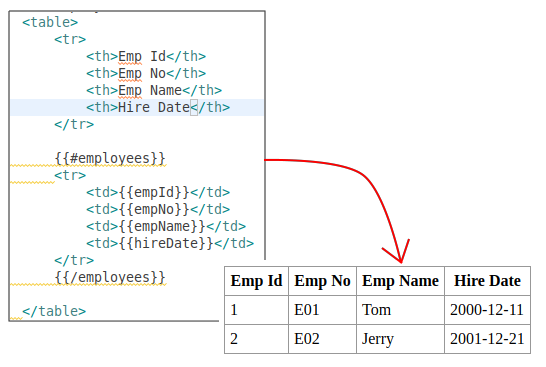
**Лабораторна робота № 20 «Розробка візуальних шарів в Spring Boot за допомогою шаблонізатора Mustache»**

Mustache це система Web Template (web шаблон), яка використовується для комбінування з даними рівня Model для створення файлів HTML . Підтримується різними мовами як Java, ActionScript, C++, Clojure, CoffeeScript, ColdFusion, PHP , ....



Mustache описується як нелогічна система (logic-less), так як вона не має чіткої команди контролю потоками програми, наприклад вона не має команди if/else (if/else statement), і не має команди циклу, натомість вона використовує теги lambda і mustache tags (тег mustache).

Наприклад, "employees " є списком, в даному контексті пари тегів відкрити & закрити : {{#employees}} & {{/employees}} будуть відповідати циклу в списку "employees" .



Назва "Mustache" (Вуса) виходить через синтаксис Mustache, який постійно використовує дужки {}, схожі на вуса.

2- Деякі приклади з Mustache

Можливо ви зустрінете деякі проблеми та непорозуміння з Mustache працюючи вперше. Нижче наведено деякі приклади, які показують дивність даної системи.

if/else?

Mustache просто не має команду умови if/else , тому вам потрібно самим створити для нього logic .

Java Code:

boolean kiss = true;

boolean hug = true;

boolean slap = false;

Mustache Template:

{{#kiss}}

Kiss ...

{{/kiss}}

{{#hug}}

Hug ...

{{/hug}}

{{#slap}}

Hic Hic ...

{{/slap}}

Результат:

Kiss ...

Hug ...

{{.}}

У Mustache {{.}} є особливим тегом (tag), подивимося на приклад :

Якщо ви маєте такі дані:

int[] numbers = **new** **int**[] {1, 2, 3, 4, 5};

String string = "Wheee!";

Mustache Template :

{{# numbers }}

\* {{ . }}

{{/ numbers }}

-----

{{# string }}

{{ . }}

{{/ string }}

Отриманий результат:

\* 1

\* 2

\* 3

\* 4

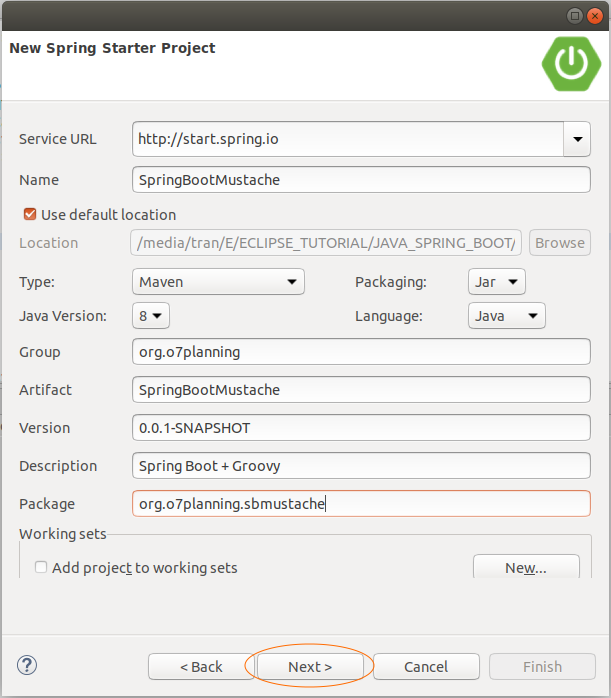
\* 5

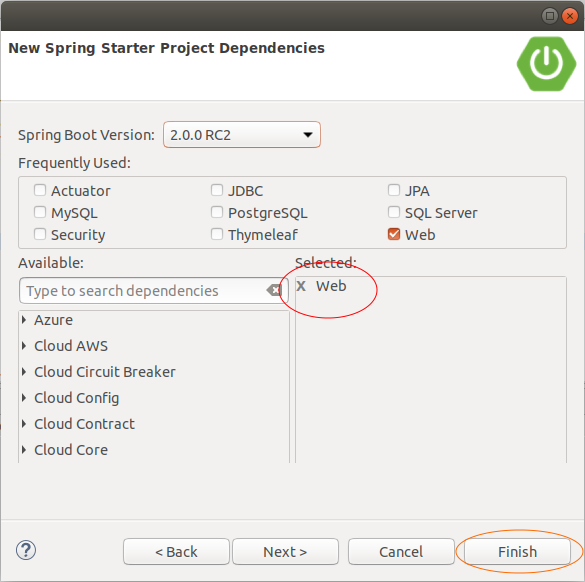
----

Wheee!

3- Створити проект Spring Boot

На Eclipse створити проект Spring Boot :





Щоб використовувати Mustache для рівня View , вам потрібно оголосити залежні spring-boot-starter-mustache у файлі pom.xml:

<**dependency**>

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

<**artifactId**>spring-boot-starter-mustache</**artifactId**>

</**dependency**>

Повний вміст файлу pom.xml :

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<**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**>

<**groupId**>org.o7planning</**groupId**>

<**artifactId**>SpringBootMustache</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**packaging**>jar</**packaging**>

<**name**>SpringBootMustache</**name**>

<**description**>Spring Boot + Groovy</**description**>

<**parent**>

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

<**artifactId**>spring-boot-starter-parent</**artifactId**>

<**version**>2.0.0.RELEASE</**version**>

<**relativePath**/> <!-- lookup parent from repository -->

</**parent**>

<**properties**>

<**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>

<**project.reporting.outputEncoding**>UTF-8</**project.reporting.outputEncoding**>

<**java.version**>1.8</**java.version**>

</**properties**>

<**dependencies**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-web</**artifactId**>

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-mustache</**artifactId**>

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-test</**artifactId**>

<**scope**>test</**scope**>

</**dependency**>

</**dependencies**>

<**build**>

<**plugins**>

<**plugin**>

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

<**artifactId**>spring-boot-maven-plugin</**artifactId**>

</**plugin**>

</**plugins**>

</**build**>

</**project**>

SpringBootMustacheApplication.java

**package** org.o7planning.sbmustache;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** **SpringBootMustacheApplication** {

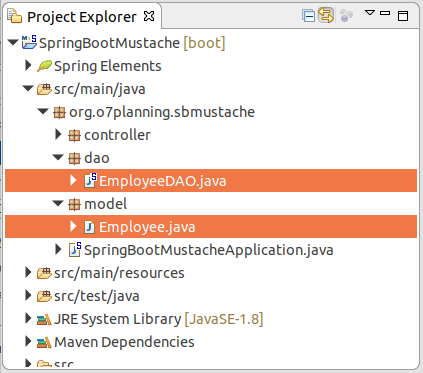
**public** **static** **void** **main**(String[] args) {

SpringApplication.run(SpringBootMustacheApplication.class, args);

}

}

4- Model, DAO



Employee.java

**package** org.o7planning.sbmustache.model;

**import** java.util.Date;

**public** **class** **Employee** {

**private** Long empId;

**private** String empNo;

**private** String empName;

**private** Date hireDate;

**public** **Employee**() {

}

**public** **Employee**(Long empId, String empNo,

String empName, Date hireDate) {

this.empId = empId;

this.empNo = empNo;

this.empName = empName;

this.hireDate = hireDate;

}

**public** Long **getEmpId**() {

**return** empId;

}

**public** **void** **setEmpId**(Long empId) {

this.empId = empId;

}

**public** String **getEmpNo**() {

**return** empNo;

}

**public** **void** **setEmpNo**(String empNo) {

this.empNo = empNo;

}

**public** String **getEmpName**() {

**return** empName;

}

**public** **void** **setEmpName**(String empName) {

this.empName = empName;

}

**public** Date **getHireDate**() {

**return** hireDate;

}

**public** **void** **setHireDate**(Date hireDate) {

this.hireDate = hireDate;

}

}

EmployeeDAO.java

**package** org.o7planning.sbmustache.dao;

**import** java.sql.Date;

**import** java.time.LocalDate;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.o7planning.sbmustache.model.Employee;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** **EmployeeDAO** {

**public** List<Employee> **getEmployees**() {

Date hireDate1= Date.valueOf(LocalDate.parse("2000-12-11"));

Employee e1= **new** **Employee**(1L, "E01", "Tom", hireDate1);

Date hireDate2= Date.valueOf(LocalDate.parse("2001-12-21"));

Employee e2= **new** **Employee**(2L, "E02", "Jerry", hireDate2);

List<Employee> list= **new** **ArrayList**<Employee>();

list.add(e1);

list.add(e2);

**return** list;

}

}

5- Controller, Mustache Template

MainController.java

**package** org.o7planning.sbmustache.controller;

**import** java.util.List;

**import** org.o7planning.sbmustache.dao.EmployeeDAO;

**import** org.o7planning.sbmustache.model.Employee;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

**public** **class** **MainController** {

@Autowired

**private** EmployeeDAO employeeDAO;

@RequestMapping("/")

**public** String **handleRequest**(Model model) {

List<Employee> employees = employeeDAO.getEmployees();

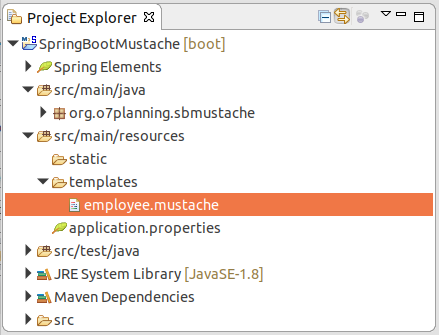
model.addAttribute("employees", employees);

**return** "employee";

}

}

Створити файл employee.mustache , розташований у файлі resources/templates проекту.



employee.mustache

<**html**>

<**head**>

<**title**>Spring Boot Mustache</**title**>

<**style**>

**table** {

**border-collapse**: collapse;

}

**table**, **td**, **th** {

**border**: 1px solid #999;

**padding**: 5px;

}

</**style**>

</**head**>

<**body**>

<**h2**>Employees</**h2**>

<**table**>

<**tr**>

<**th**>Emp Id</**th**>

<**th**>Emp No</**th**>

<**th**>Emp Name</**th**>

<**th**>Hire Date</**th**>

</**tr**>

{{#employees}}

<**tr**>

<**td**>{{empId}}</**td**>

<**td**>{{empNo}}</**td**>

<**td**>{{empName}}</**td**>

<**td**>{{hireDate}}</**td**>

</**tr**>

{{/employees}}

</**table**>

</**body**>

</**html**>

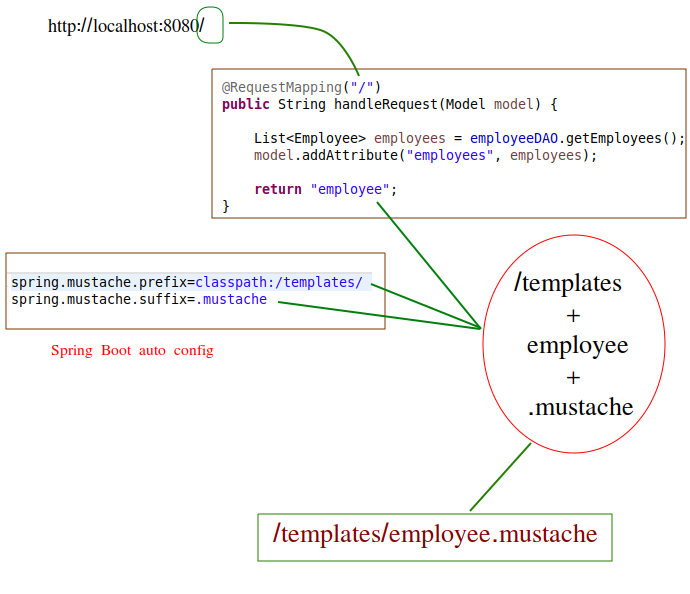
Spring Boot 2.x автоматично конфігурує Mustache з наступними властивостями за промовчанням:

application.properties (Default by Spring Boot)

spring.mustache.prefix=classpath:/templates/

spring.mustache.suffix=.mustache

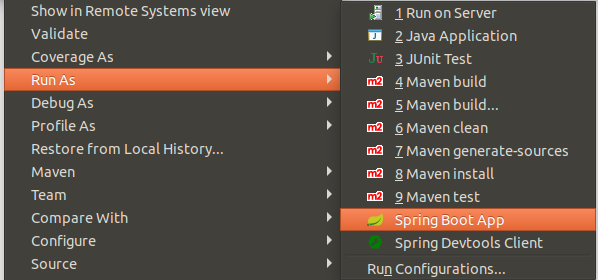
Зображення описує потік програми та зв'язок між Controller та Mustache View :



6- Запуск програми

Натиснути на праву кнопку миші на проект, вибрати:

* Run As/Spring Boot App



Після цього, пройти за посиланням:

* [http://localhost:8080](http://localhost:8080/)

