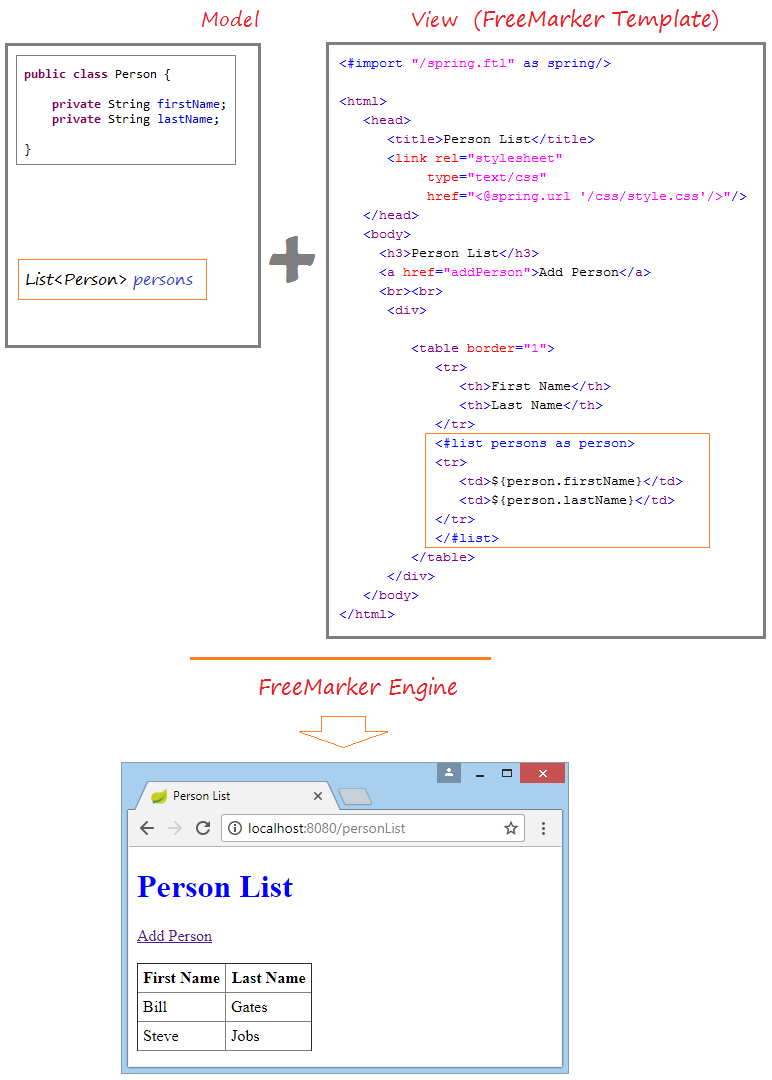
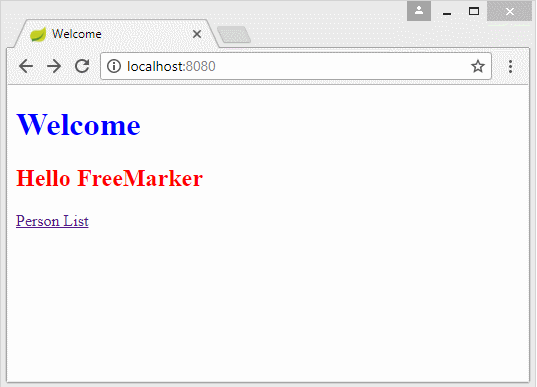
**Лабораторна робота № 18 «Розробка візуальних шарів в Spring Boot за допомогою шаблонізатора Freemarker»**

FreeMarker - це Template Engine (Шаблонізатор), який надається за допомогою Apache як вихідний ресурс бібліотеки Java . FreeMarker читає файли шаблонів і поєднує їх з об’єктами Java , щоб згенерувати (згенерувати) вихідний текст (HTML, електронна пошта).

Шаблонні файли (файл шаблону) FreeMarker насправді є текстовими файлами у форматі, представленому за допомогою FreeMarker , це мова шаблонів FreeMarker (FTL).



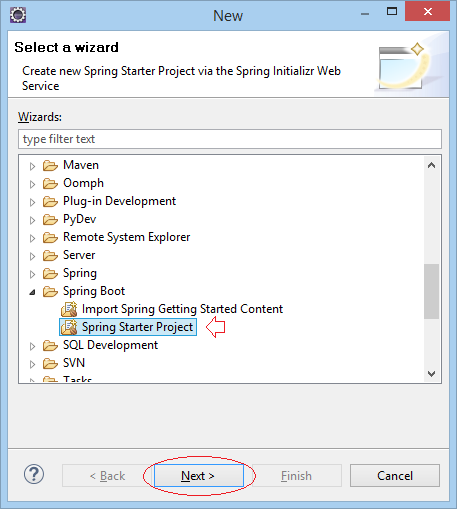
FreeMarker може бути використаний для заміни JSP на рівні View (View Layer) додаток Web MVC. Нижче є зображення додатка, яке ми виповнюємо в цій статті:



2- Створити додаток

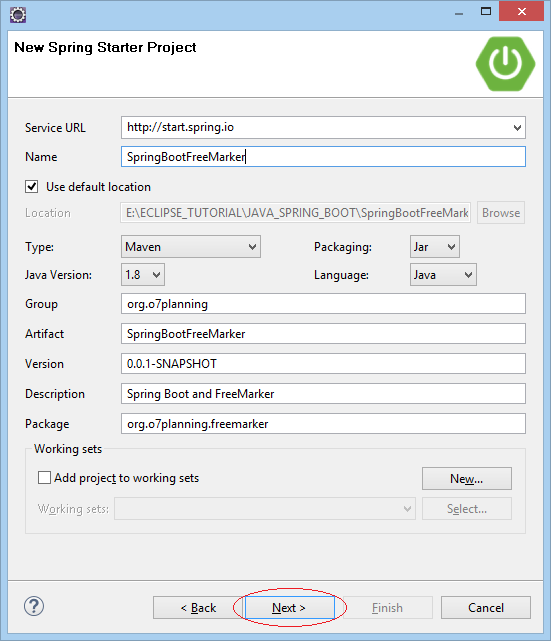
У Eclipse ви можете:

* Файл/Новий/Інше...

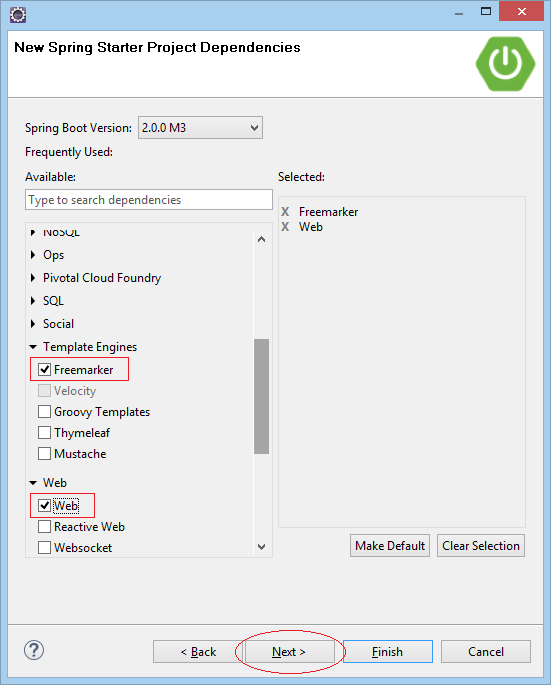


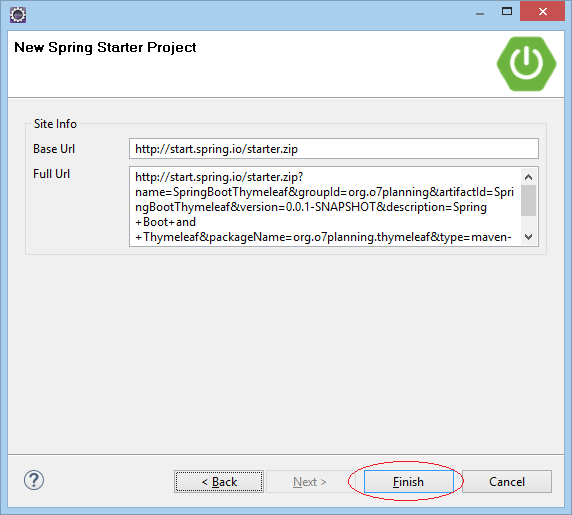
Введіть:

* Назва : SpringBootFreeMarker
* Група : org.o7planning
* Артефакт : SpringBootFreeMarker
* Опис : Spring Boot і FreeMarker
* Пакет : org.o7planning.freemarker

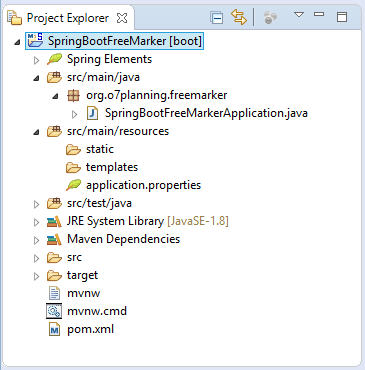


Виберіть 2 технології Web і FreeMarker .





Ваш проект створено:



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**>SpringBootFreeMarker</**artifactId**>

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

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

<**name**>SpringBootFreeMarker</**name**>

<**description**>Spring Boot and FreeMarker</**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-freemarker</**artifactId**>

</**dependency**>

<**dependency**>

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

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

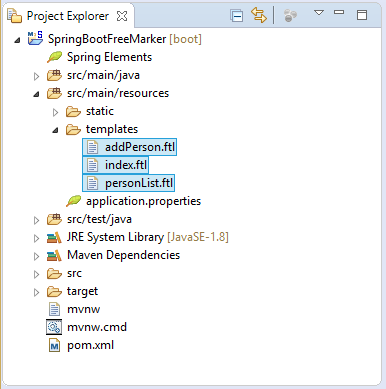
</**plugins**>

</**build**>

</**project**>

3- Шаблон FreeMarker

Ми створили 3 файли шаблону (файл шаблону) і розташували в папці src/main/resources/templates :



index.flt

<#import "/spring.ftl" as spring/>

<!DOCTYPE **HTML**>

<**html**>

<**head**>

<**meta** charset="UTF-8" />

<**title**>Welcome</**title**>

<**link** rel="stylesheet"

type="text/css" href="<@spring.url '/css/style.css'/>"/>

</**head**>

<**body**>

<**h1**>Welcome</**h1**>

<#if message??>

<**h2**>${message}</**h2**>

</#if>

<**a** href="<@spring.url '/personList'/>">Person List</**a**>

</**body**>

</**html**>

personList.flt

<#import "/spring.ftl" as spring/>

<**html**>

<**head**>

<**title**>Person List</**title**>

<**link** rel="stylesheet"

type="text/css"

href="<@spring.url '/css/style.css'/>"/>

</**head**>

<**body**>

<**h3**>Person List</**h3**>

<**a** href="addPerson">Add Person</**a**>

<**br**><**br**>

<**div**>

<**table** border="1">

<**tr**>

<**th**>First Name</**th**>

<**th**>Last Name</**th**>

</**tr**>

<#list persons as person>

<**tr**>

<**td**>${person.firstName}</**td**>

<**td**>${person.lastName}</**td**>

</**tr**>

</#list>

</**table**>

</**div**>

</**body**>

</**html**>

addPerson.flt

<#import "/spring.ftl" as spring/>

<**html**>

<**head**>

<**title**>Add Person</**title**>

<**link** rel="stylesheet"

type="text/css" href="<@spring.url '/css/style.css'/>"/>

</**head**>

<**body**>

<#if errorMessage??>

<**div** style="color:red;font-style:italic;">

${errorMessage}

</**div**>

</#if>

<**div**>

<**fieldset**>

<**legend**>Add Person</**legend**>

<**form** name="person" action="" method="POST">

First Name: <@spring.formInput "personForm.firstName" "" "text"/> <**br**/>

Last Name: <@spring.formInput "personForm.lastName" "" "text"/> <**br**/>

<**input** type="submit" value="Create" />

</**form**>

</**fieldset**>

</**div**>

</**body**>

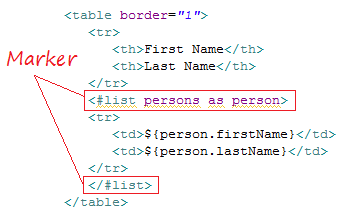
</**html**>

Усі файли шаблонів повинні містити "Макроси FreeMarker" .

<!-- FreeMarker Macros -->

<#import "/spring.ftl" as spring/>

У шаблонних файлах (Template file) є FreeMarker Marker (Метки FreeMarker), вони є інструкцією, що допомагає FreeMarker Engine обробляти дані. FreeMarker Engine аналізує шаблонний файл (файл шаблону) і поєднується з даними Java , щоб створити (згенерувати) новий документ.



Нижче наведено приклади використання Context-Path у FreeMarker :

<!-- Example 1: -->

<**a** href="<@spring.url '/mypath/abc.html'/>">A Link</**a**>

Output: ==>

<**a** href="/my-context-path/mypath/abc.html">A Link</**a**>

<!-- Example 2: -->

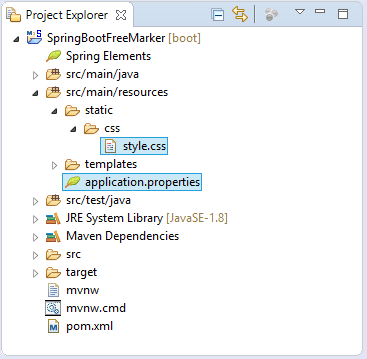
<**form** action="<@spring.url '/mypath/abc.html'/>" method="POST">

Output: ==>

<**form** action="/my-context-path/mypath/abc.html" method="POST">

4- Статичний файл ресурсів і властивостей

Такими статичними ресурсами (Static Resource), наприклад файлами css , javascript , image , вам потрібно розташувати їх у папці src/main/resources/static або в її підпапках.



style.css

**h1** {

**color**:#0000FF;

}

**h2** {

**color**:#FF0000;

}

**table** {

**border-collapse**: collapse;

}

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

**padding**: 5px;

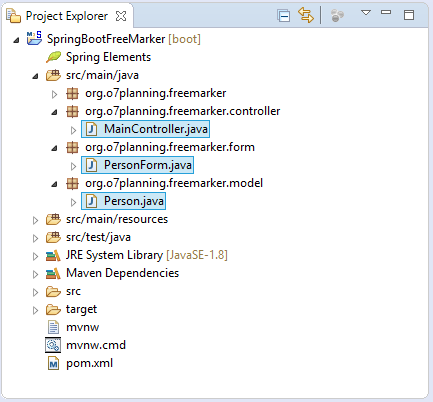
}

додаток.властивості

welcome.message=Hello FreeMarker

error.message=First Name & Last Name is required!

5- Класи, Модель, Форма, Контролер



Person.java

**package** org.o7planning.freemarker.model;

**public** **class** **Person** {

**private** String firstName;

**private** String lastName;

**public** **Person**() {

}

**public** **Person**(String firstName, String lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

**public** String **getFirstName**() {

**return** firstName;

}

**public** **void** **setFirstName**(String firstName) {

this.firstName = firstName;

}

**public** String **getLastName**() {

**return** lastName;

}

**public** **void** **setLastName**(String lastName) {

this.lastName = lastName;

}

}

Клас PersonForm містить дані FORM , коли ви створюєте новий Person на сторінці addPerson .

PersonForm.java

**package** org.o7planning.freemarker.form;

**public** **class** **PersonForm** {

**private** String firstName;

**private** String lastName;

**public** String **getFirstName**() {

**return** firstName;

}

**public** **void** **setFirstName**(String firstName) {

this.firstName = firstName;

}

**public** String **getLastName**() {

**return** lastName;

}

**public** **void** **setLastName**(String lastName) {

this.lastName = lastName;

}

}

MainController є класом Controller , він обробляє запити користувача та контролює потік (потік) програми.

MainController.java

**package** org.o7planning.freemarker.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.o7planning.freemarker.form.PersonForm;

**import** org.o7planning.freemarker.model.Person;

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

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

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

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

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

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

@Controller

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

**private** **static** List<Person> persons = **new** **ArrayList**<Person>();

**static** {

persons.add(**new** **Person**("Bill", "Gates"));

persons.add(**new** **Person**("Steve", "Jobs"));

}

// Інжектувати (inject) з application.properties.

@Value("${welcome.message}")

**private** String message;

@Value("${error.message}")

**private** String errorMessage;

@RequestMapping(value = { "/", "/index" }, method = RequestMethod.GET)

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

model.addAttribute("message", message);

**return** "index";

}

@RequestMapping(value = { "/personList" }, method = RequestMethod.GET)

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

model.addAttribute("persons", persons);

**return** "personList";

}

@RequestMapping(value = { "/addPerson" }, method = RequestMethod.GET)

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

PersonForm personForm = **new** **PersonForm**();

model.addAttribute("personForm", personForm);

**return** "addPerson";

}

@RequestMapping(value = { "/addPerson" }, method = RequestMethod.POST)

**public** String **addPersonSave**(Model model, //

@ModelAttribute("personForm") PersonForm personForm) {

String firstName = personForm.getFirstName();

String lastName = personForm.getLastName();

**if** (firstName != null && firstName.length() > 0 //

&& lastName != null && lastName.length() > 0) {

Person newPerson = **new** **Person**(firstName, lastName);

persons.add(newPerson);

**return** "redirect:/personList";

}

String error = "First Name & Last Name is required!";

model.addAttribute("errorMessage", error);

**return** "addPerson";

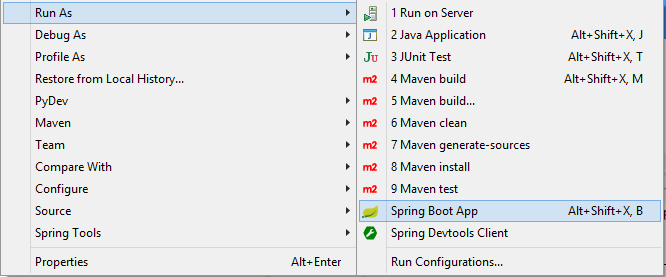
}

}

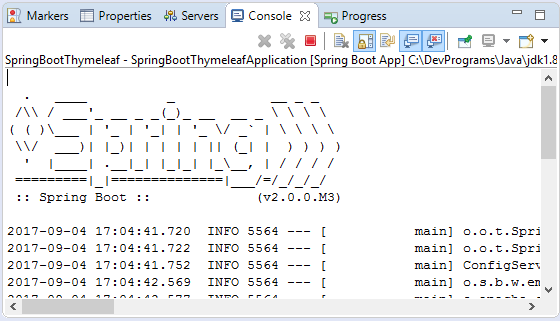
6- Запуск додатків

Щоб запустити додаток, натисніть праву кнопку миші на Project, виберіть:

* Запустіть додаток As/Spring Boot.



Додаток розвернуто (розгорнуто) на вбудованому веб-сервері.



Запустіть наступну URL-адресу в браузері:

* <http://localhost:8080/>

