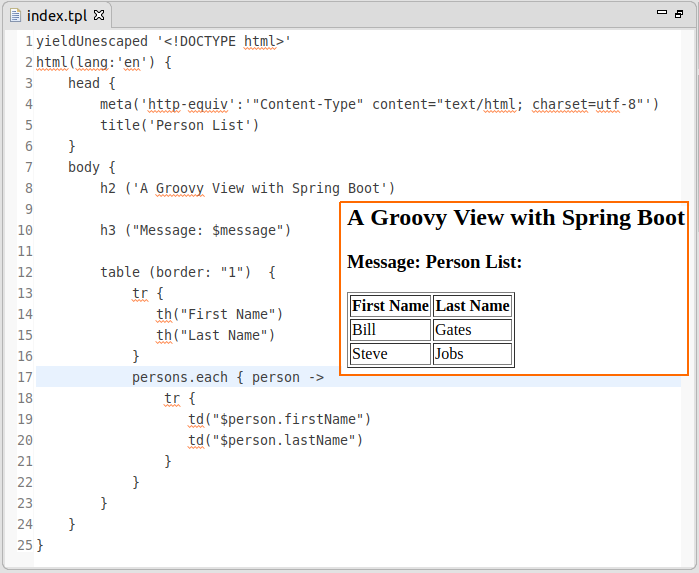
**Лабораторна робота № 19 «Розробка візуальних шарів в Spring Boot**

**за допомогою Groovy»**

**Apache Groovy** — це об’єктно-орієнтована мова програмування (мова об’єктно-орієнтованого програмування), що працює на платформі **Java** . Він є динамічною мовою (динамічна мова) з такими функціями, як **Python, Ruby, Perl** . Він може використовуватися в якості мовного сценарію для платформи **Java**, і скомпільований у байкод коду **віртуальної** машини **Java**, і взаємодіє з іншими кодами **Java** і бібліотеками. **Groovy** використовує синтаксис скобки {} як **Java**. Майже всі коди **Java** мають дійсний синтаксис із **Groovy**, хоча семантика може бути іншою.

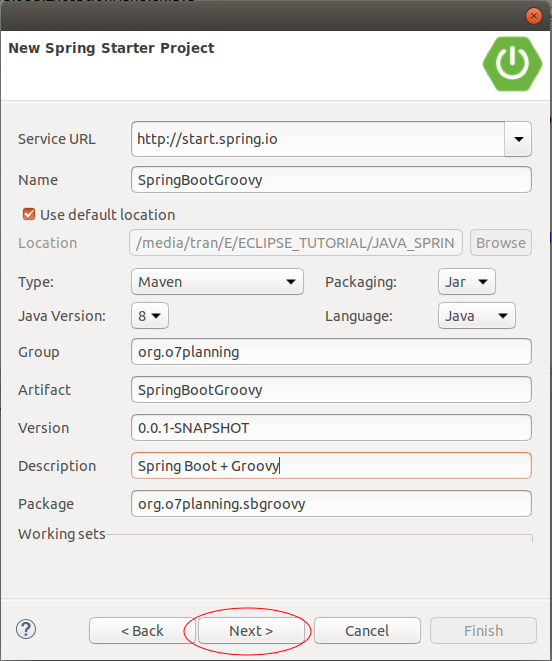
У цій статті я не фокусуюся на введенні в мову **Groovy** як і про використання цієї мови. Але **Groovy** надає шаблон (шаблон), щоб створити документи **HTML**, і цю тему ми будемо обговорювати в статті.

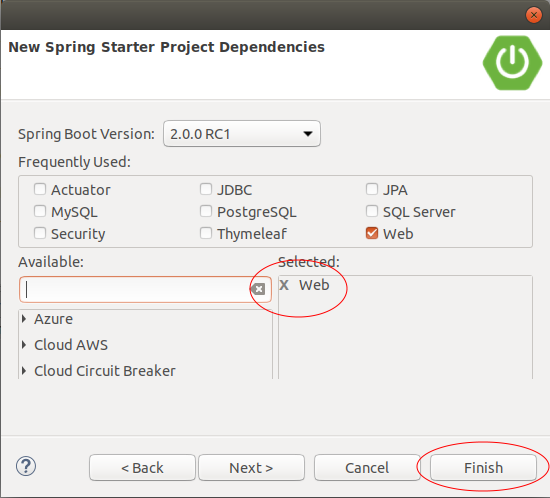


Ціль цієї статті створити **Java Web** за допомогою **Spring Boot** і за допомогою **шаблону Groovy** для рівня **View.**  Звичайно для рівня **View** ви можете вибрати інші технології як **JSP** , **Thymeleaf**.

## 2- Створити проект Spring Boot

У **Eclipse** створити проект **Spring Boot** :





Щоб використовувати **Groovy,** вам потрібно додати залежні **шаблони spring-boot-starter-groovy** у ваш проект.

<**dependency**>

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

<**artifactId**>spring-boot-starter-groovy-templates</**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**>SpringBootGroovy</**artifactId**>

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

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

<**name**>SpringBootGroovy</**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-groovy-templates</**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**>

SpringBootGroovyApplication.java

**package** org.o7planning.sbgroovy;

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

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

@SpringBootApplication

**public** **class** **SpringBootGroovyApplication** {

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

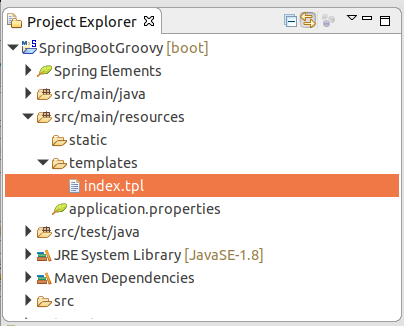
SpringApplication.run(SpringBootGroovyApplication.class, args);

}

}

## 3- Контролер, шаблон Groovy

Створіть файл **index.tpl** в папці **templates** .



index.tpl

yieldUnescaped '<!DOCTYPE html>'

html(lang:'en') {

head {

meta('http-equiv':'"Content-Type" content="text/html; charset=utf-8"')

title('Person List')

}

body {

h2 ('A Groovy View with Spring Boot')

h3 ("Message: $message")

table (border: "1") {

tr {

th("First Name")

th("Last Name")

}

persons.each { person ->

tr {

td("$person.firstName")

td("$person.lastName")

}

}

}

}

}

MainController.java

**package** org.o7planning.sbgroovy.controller;

**import** java.util.ArrayList;

**import** java.util.List;

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

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

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

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

@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"));

}

@RequestMapping(value = "/")

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

String message = "Person List:";

model.addAttribute("message", message);

model.addAttribute("persons", persons);

**return** "index";

}

}

Person.java

**package** org.o7planning.sbgroovy.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;

}

}

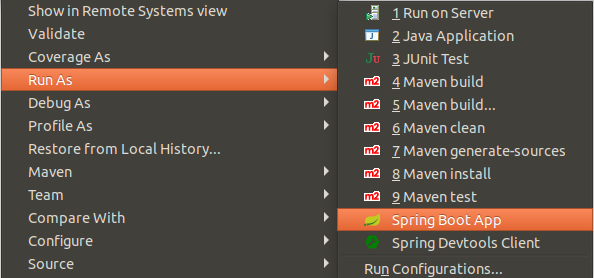
Зображення нижче пояснює відношення між **контролером** і **Groovy View** :

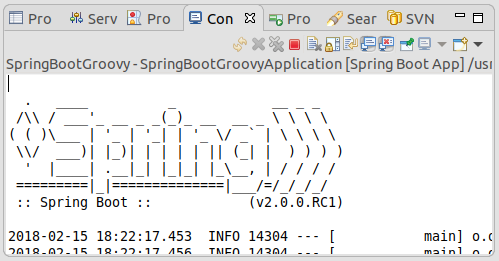
### Зображення TODO.

## 4- Запуск додатків

Нажати на праву кнопку миші на проект і вибрати:

* **Запустіть додаток As/Spring Boot**





Тепер ваше додаток запущено. У браузері перейдіть до наступної посилки:

* <http://localhost:8080/>

