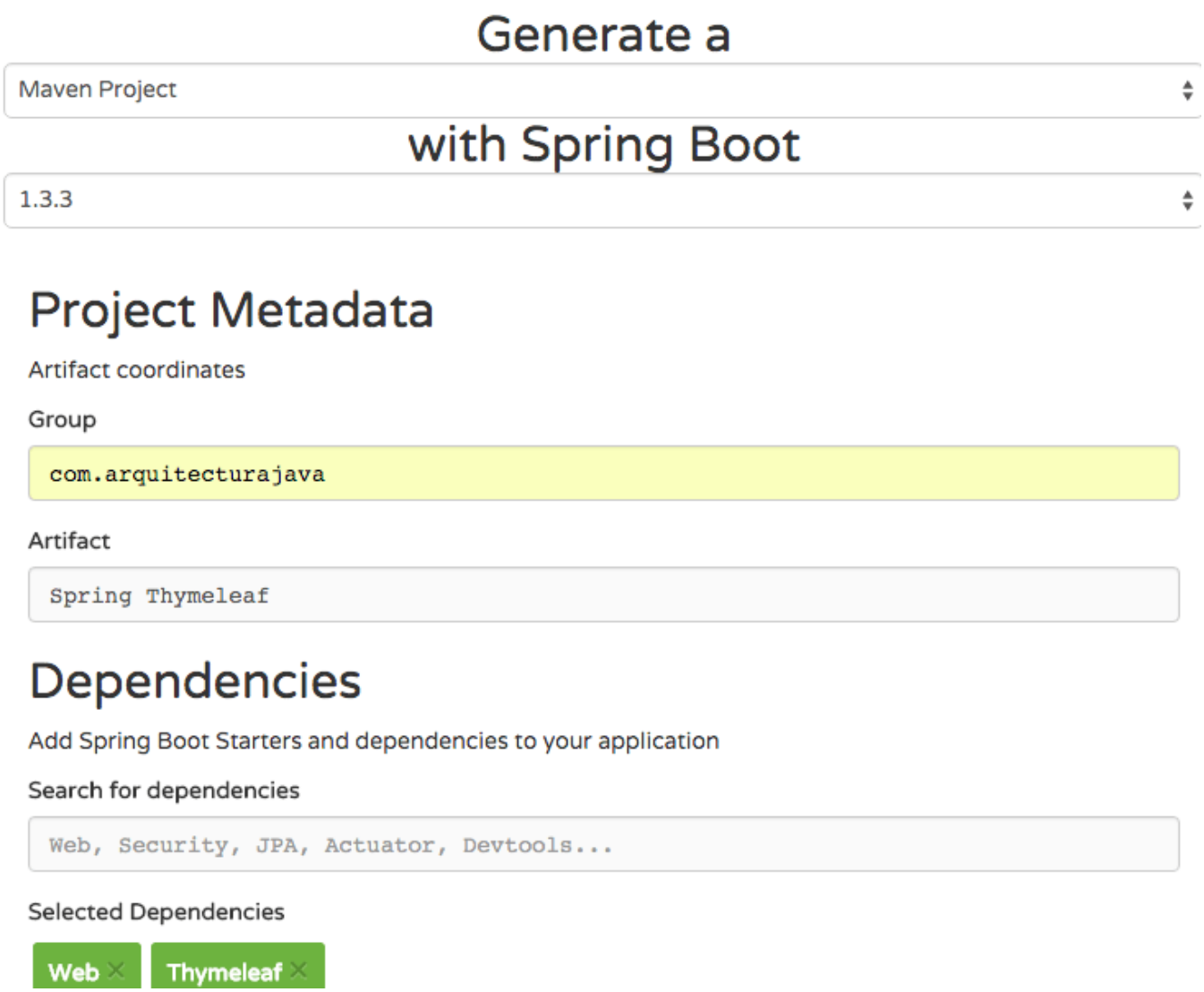
Thymeleaf es un nuevo motor de plantillas que se usa mucho con Spring MVC . Durante muchos años hemos utilizado librerías de etiquetas para gestionar la capa de presentación tipo JSTL o Spring Forms ,pero el uso de etiquetas aunque potente no resulta del todo natural. Los motores de plantillas siempre han sido una tecnología interesante y se han usado ampliamente soluciones como Velocity o FreeMaker . Thymeleaf retoma este enfoque y lo potencia. Vamos a ver como configurar este motor de plantillas seleccionando las dependencias con SpringBoot

---



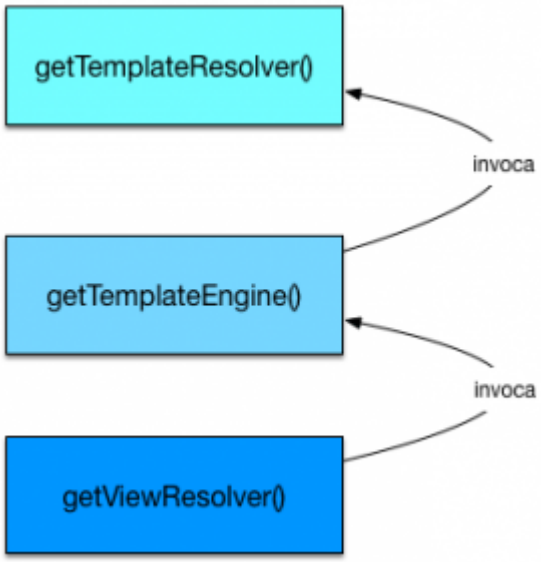
Configuracion Thymeleaf

En este caso utilizamos las dependencias de Web y de Thymeleaf que es el motor de plantillas . El siguiente paso que tenemos que realizar es con el proyecto que nos hemos descargado configurar el motor de plantillas. Ese paso se realiza en el fichero de aplicación que viene en el proyecto.

```
package com.arquitecturajava;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.thymeleaf.spring4.SpringTemplateEngine;
import org.thymeleaf.spring4.view.ThymeleafViewResolver;
import org.thymeleaf.templateresolver.ServletContextTemplateResolver;

@SpringBootApplication
@ComponentScan
public class ThymeleafApplication {

 @Bean(name ="templateResolver")
 public ServletContextTemplateResolver getTemplateResolver() {
 ServletContextTemplateResolver templateResolver = new
ServletContextTemplateResolver();
 templateResolver.setPrefix("/WEB-INF/templates/");
 templateResolver.setSuffix(".html");
 templateResolver.setTemplateMode("XHTML");
 return templateResolver;
```

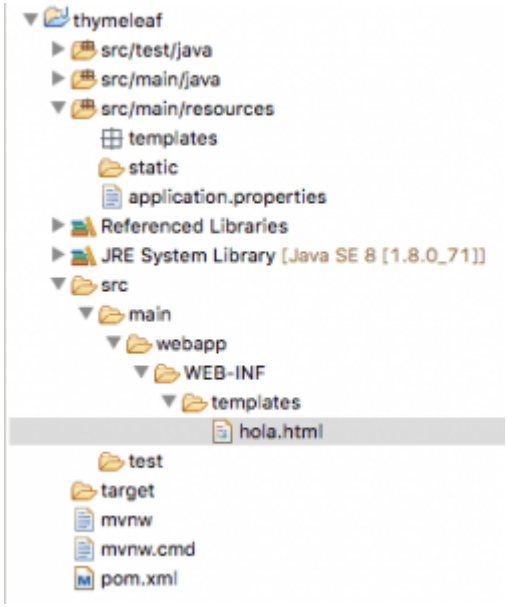
```
 }
 @Bean(name ="templateEngine")
 public SpringTemplateEngine getTemplateEngine() {
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();
 templateEngine.setTemplateResolver(getTemplateResolver());
 return templateEngine;
 }
 @Bean(name="viewResolver")
 public ThymeleafViewResolver getViewResolver(){
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
 viewResolver.setTemplateEngine(getTemplateEngine());
 return viewResolver;
 }


 public static void main(String[] args) {
 SpringApplication.run(ThymeleafApplication.class, args);
 }
}
```

Acabamos de utilizar la sintaxis de anotaciones para configurar el motor de plantillas configurando tres beans.



Puede parecer complicado en un primer momento pero realmente son tres pasos sencillos de entender.

1. El primer bean define donde se encuentran las plantillas
2. El segundo bean que motor de plantillas se va a utilizar
3. El tercero se apoya en los anteriores para configurar el resolutor de vistas y redirigirnos a la vista adecuada.
   El siguiente paso a realizar es crear una plantilla dentro de nuestro proyecto en la carpeta especificada.



El fichero de hola.html contendrá el código del lenguaje de plantillas de Spring ThymeLeaf.

```
<html>
<body>
 <span th:text="${persona.nombre}"></span>
 <span th:text="${persona.apellidos}"></span>
</body>
</html>
```

La plantilla mostrará el nombre y apellidos de una persona .Vamos a crear la clase:

```
package com.arquitecturajava;

public class Persona {

private String nombre;
```

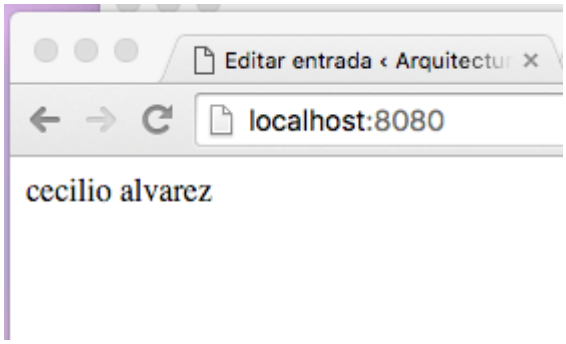
```
private String apellidos;
public String getNombre() {
return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;
}
public String getApellidos() {
return apellidos;
}
public void setApellidos(String apellidos) {
this.apellidos = apellidos;
}
public Persona(String nombre, String apellidos) {
super();
this.nombre = nombre;
this.apellidos = apellidos;
}

}
```

El último paso es crear un controlador que cargue los datos en la plantilla:

```
package com.arquitecturajava;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
@Controller
public class ControladorHola {
@RequestMapping("/")
String home(Model modelo) {

modelo.addAttribute("persona", new Persona("cecilio","alvarez"));

return "hola";
}
}
```

Desplegamos la aplicación y vemos el resultado por pantalla:



Acabamos de construir nuestro primer ejemplo con el motor de ThymeLeaf.

Otros artículos relacionados: Modelo MVC cliente y servidor , Introducción a Spring MVC , Spring Security