Spring @Controller es una de las anotaciones más habituales de Spring Framework , todo el mundo ha tenido en algún momento que crear un controlador .Los controladores sirven para comunicar información entre la vista y el modelo . Es decir yo por ejemplo puedo tener una lista de Personas a nivel del Modelo y quiero pasarlo a la vista y que la vista se encargue de mostrar la información que tenemos en el modelo. Para ello lo primero que tenemos que hacer es crear un proyecto de Spring Boot que contenga el Starter Web y el de ThymeLeaf por lo menos.

```
<dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
```

Una vez que tenemos estas dependencias construidas , podemos crearnos una clase de negocio Persona.

```
package com.arquitecturajava.web1.models;

import java.util.Objects;

public class Persona {

        private String nombre;
        private String apellidos;
        private int edad;
        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 int getEdad() {
        return edad;
}
public void setEdad(int edad) {
        this.edad = edad;
}
public Persona(String nombre, String apellidos, int edad) {
        super();
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
}
public Persona() {
        super();
}
@Override
public int hashCode() {
        return Objects.hash(nombre);
}
@Override
```

```
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Persona other = (Persona) obj;
		return Objects.equals(nombre, other.nombre);
	}
	public Persona(String nombre) {
		super();
		this.nombre = nombre;
	}
}
```

Spring @Controller

Una vez que tenemos construida la clase es momento de construir un Spring @Controller que es el encargado de ligar la url de /personas/lista con los datos que tenemos en estos momentos en memoria

```
package com.arquitecturajava.web1.controllers;

import java.util.ArrayList;
import java.util.List;

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

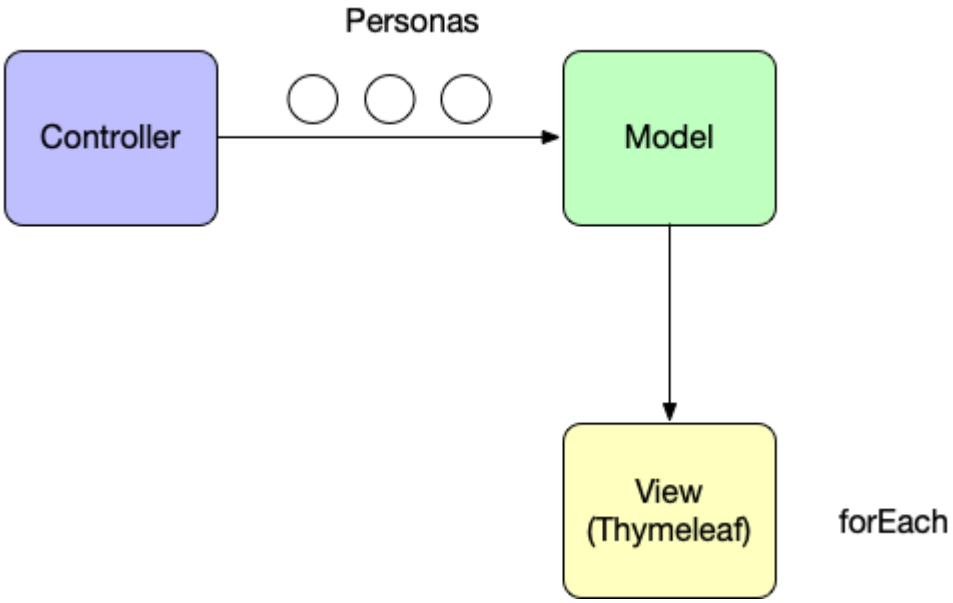
```
import org.springframework.web.bind.annotation.RequestParam;

import com.arquitecturajava.web1.models.Persona;

@Controller
@RequestMapping("/personas")
public class PersonaController {

	static List<Persona> lista= new ArrayList<Persona>();
	static {
		lista.add(new Persona ("pepe","perez",20));
		lista.add(new Persona ("ana","gomez",40));
	}
	@GetMapping("/lista")
	public String lista(Model modelo) {
		modelo.addAttribute("lista", lista);
		return "lista";
	}
}
```

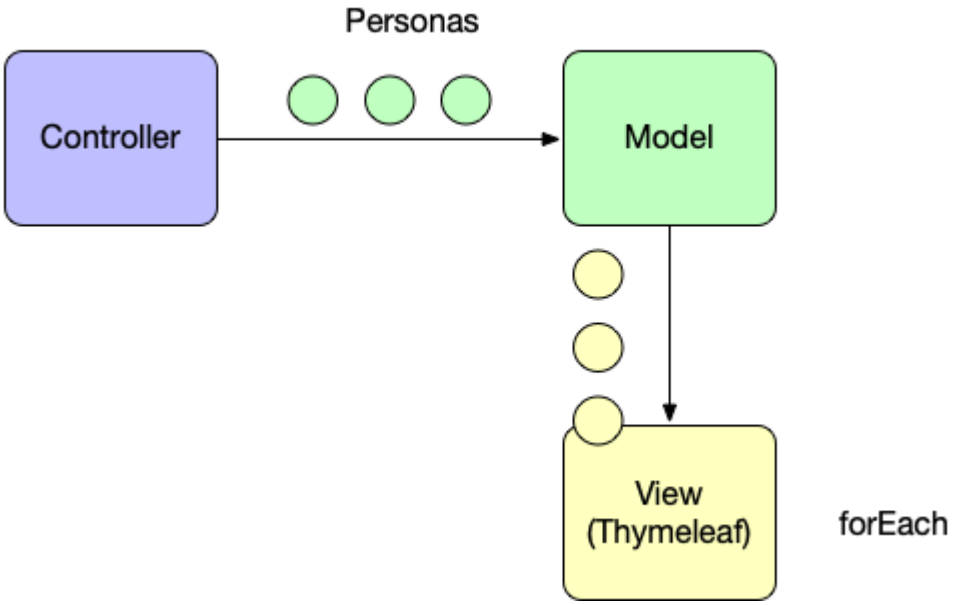
Como se puede ver el Controlador usa una variable de tipo Model para comunicar datos entre el propio controlador y la vista . Esta comunicación se hace a través del método addAttribute que permite comunicar ambas partes .



La vista recibe los datos y tendrá que ser encargada de mostrarlos.

ThymeLeaf Plantillas

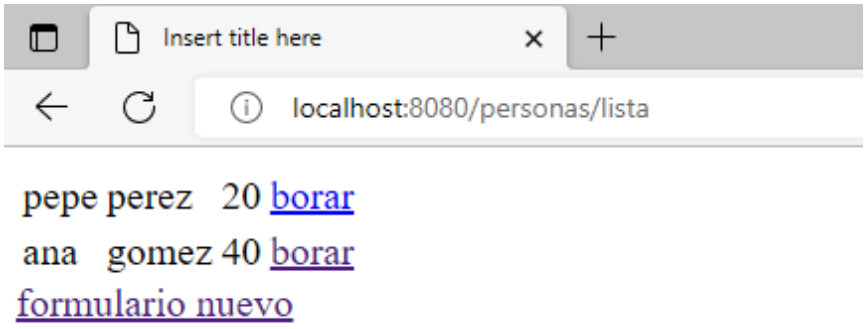
La vista se encarga de recibir los datos y a traves de ThymeLeaf y el motor de plantillas las renderiza.



Es momento de ver el código de la plantilla para entender como funciona el motor:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" >
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
	<table>
		<tr th:each="persona : ${lista}">
			<td th:text="${persona.nombre}"></td>
			<td th:text="${persona.apellidos}"></td>
			<td th:text="${persona.edad}"></td>
			</tr>
	</table>
	<a th:href="@{formulario}">formulario nuevo</a>
</body>
</html>
```

La plantilla usa un bucle each para recorrer cada una de las personas e imprimir su resultado para ello usara también la propiedad text que indica que texto queremos que muestre en cada celda de la tabla generada . Una vez configuradas estas cosas es momento de cargar la página.



Spring @Controller es una de las anotaciones más comunes en aplicaciones web clásicas MPA

Otros artículos relacionados

- Spring 5 Reactive y Thymeleaf
- ¿Qué es un Maven Property y como se utiliza?
- Spring @Service , usando el patrón Servicio
- Spring @Autowired y la inyección de dependencias
- Usando @SpringBootApplication una anotación clave
- Spring Controller