Sistemas Distribuidos

Bloque II

Desarrollo de sistemas distribuidos y aplicaciones web.

Tema 2.1:

Spring Framework

URJC Escuela Técnica Superior de Ingeniería Informática

Nicolás Rodríguez
nicolas.rodriguez@urjc.es

Índice

Definición y origen

- Spring Boot es un framework que se utiliza para desarrollar aplicaciones basadas en Java.
- Funciona como una extensión del ya conocido Spring Framework, pero con un enfoque en simplificar el proceso de configuración y despliegue de aplicaciones.
- Es ideal para construir tanto aplicaciones web como servicios de backend.

Índice

- Definición y origen
- **Características principales**
- Ecosistema
- MVC
- Maven
- Configuración de un proyecto básico
- Ejemplo

Características principales

- Reducción de Configuración:
  - Uno de los principales atractivos de Spring Boot es su capacidad para minimizar la configuración requerida para arrancar una aplicación Spring.
- Enfoque en la Convención:
  - Spring Boot sigue el principio de "convenio sobre configuración", lo que significa que intenta adivinar la configuración que necesitas en función de las bibliotecas que tienes en tu classpath.
- Relación con Spring:
  - Ofrece una configuración predeterminada que ayuda a reducir el tiempo de arranque y desarrollo.
- Ventaja adicional:
  - Capacidad de crear aplicaciones independientes.

Índice

- Definición y origen
- Características principales
- **Ecosistema**
- MVC
- Maven
- Configuración de un proyecto básico
- Ejemplo

Ecosistema

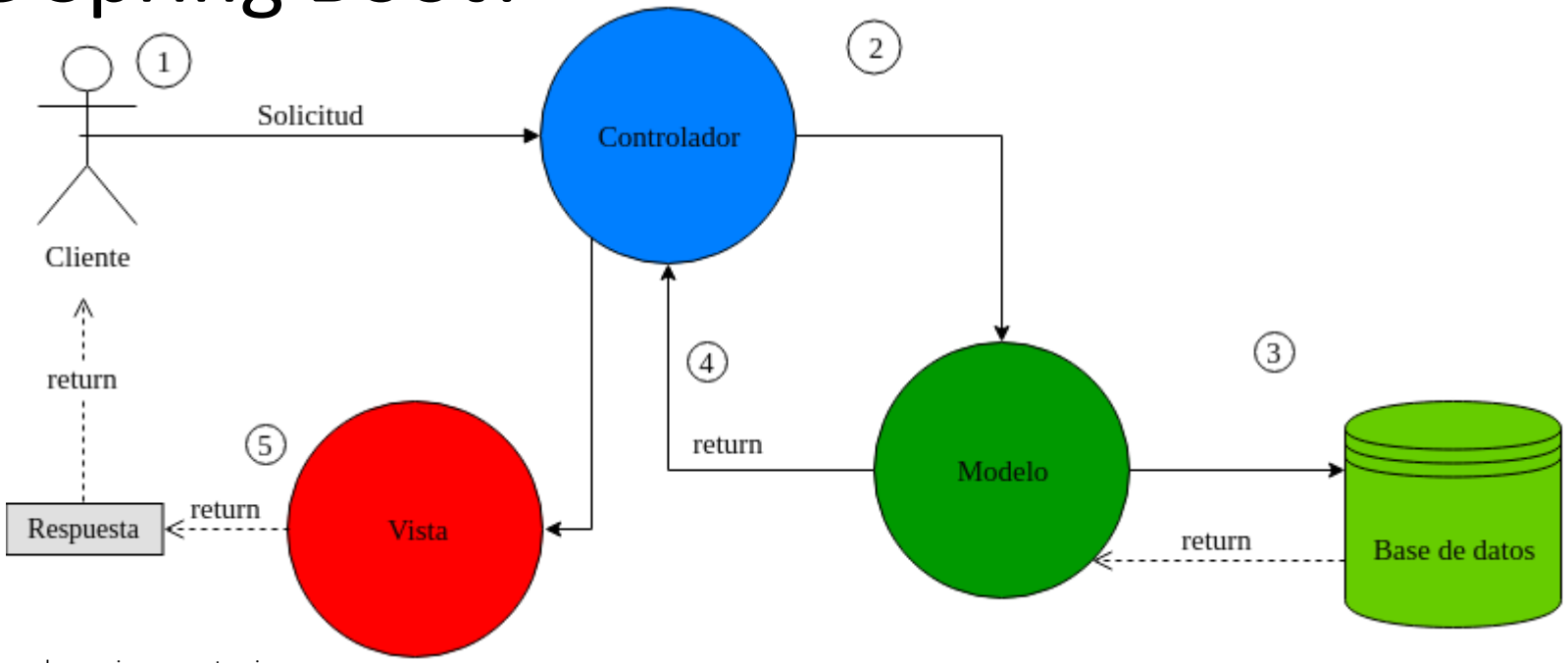
- Spring Boot es un miembro del ecosistema de Spring, que incluye proyectos como Spring Framework, Spring Data, Spring Security, entre otros.
- Se usa Spring MVC para desarrollar aplicaciones web, y Spring Boot simplifica significativamente la configuración requerida.
- Despliegue Independiente y microservicios: facilita la creación de aplicaciones independientes y su relevancia en la construcción de arquitecturas de microservicios.

Índice

- Definición y origen
- Características principales
- Ecosistema
- **MVC**
- Maven
- Configuración de un proyecto básico
- Ejemplo

MVC

- El modelo Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software ampliamente utilizado en el desarrollo de aplicaciones web.
- ¿Cómo funciona cada componente de este patrón en el contexto de Spring Boot?



Fuente: https://github.com/DavidBrionesFF/planilla-empleados-sql-server-jasperreport-spring

MVC

- El Modelo en una aplicación Spring Boot representa la capa de datos y la lógica de negocio. Consiste en objetos que llevan datos y la lógica relacionada con estos datos. Estos objetos suelen ser POJOs (Plain Old Java Objects) que representan entidades, es decir, reflejan las tablas en una base de datos.
- En Spring Boot, el modelo se gestiona típicamente mediante JPA (Java Persistence API) para mapear estos objetos a registros de base de datos. Spring Data JPA se utiliza a menudo para simplificar las operaciones de base de datos, proporcionando una interfaz de repositorio que maneja tareas comunes (CRUD)

MVC

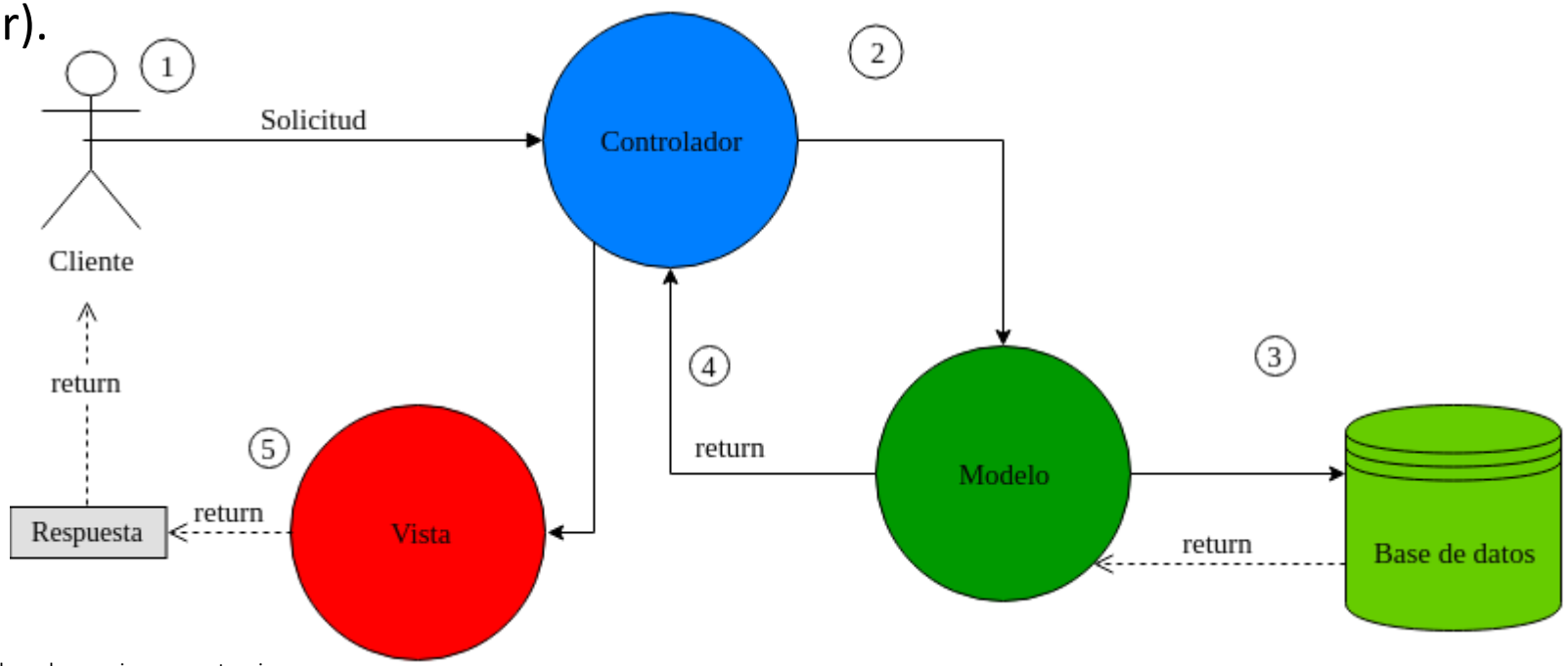
- La Vista es responsable de presentar los datos al usuario. En el contexto de una aplicación web, esto implica generar HTML, CSS y JavaScript para mostrar en el navegador del usuario.
- Spring Boot utiliza plantillas para generar la vista. Estas plantillas pueden ser Thymeleaf, Mustache, JSP, o cualquier otra tecnología de plantillas compatible. Estas plantillas definen cómo se presenta la información del modelo al usuario, y Spring Boot las procesa para generar el HTML final que se envía al navegador.

MVC

- El Controlador actúa como intermediario entre la Vista y el Modelo. Maneja las solicitudes entrantes del usuario (generalmente solicitudes HTTP), interactúa con el modelo para procesar datos o realizar operaciones de negocio, y luego elige una vista para presentar la salida.
- En Spring Boot, los controladores se implementan como clases anotadas con @Controller o @RestController. Estas clases utilizan anotaciones como @RequestMapping o variantes (@GetMapping, @PostMapping, etc.) para mapear diferentes acciones a métodos específicos. En el caso de @RestController, se utiliza principalmente para servicios API donde la respuesta es generalmente JSON o XML, en lugar de una vista HTML.

MVC

- Flujo de trabajo de Solicitud/Respuesta:
  - Cuando un usuario realiza una solicitud (por ejemplo, al cargar una página o enviar un formulario), esta solicitud es manejada por un controlador.
  - El controlador procesa la solicitud, interactúa con el modelo si es necesario (por ejemplo, para recuperar datos o actualizar la base de datos), y luego devuelve una respuesta.
  - Esta respuesta puede ser una vista renderizada (en caso de @Controller) o datos (en caso de @RestController).



Fuente: https://github.com/DavidBrionesFF/planilla-empleados-sql-server-jasperreport-spring

Índice

- Definición y origen
- Características principales
- Ecosistema
- MVC
- **Maven**
- Configuración de un proyecto básico
- Ejemplo

Maven

- Maven es una herramienta de gestión de proyectos y comprensión de software utilizada para manejar dependencias, compilar el proyecto y gestionar el ciclo de vida del software. En proyectos Spring Boot, Maven es comúnmente utilizado para configurar dependencias, plugins y otras configuraciones del proyecto.
- El archivo pom.xml en un proyecto Spring Boot es el corazón de la configuración de Maven. En él se definen las dependencias del proyecto, la versión de Spring Boot utilizada, plugins y cualquier otra configuración específica de Maven necesaria para el proyecto.

Configuración de un proyecto básico

- Spring Initializr (disponible en start.spring.io) es una herramienta en línea que facilita la generación de un proyecto Spring Boot. Se puede seleccionar la versión de Spring Boot, las dependencias iniciales (como Web, JPA, Thymeleaf, etc.), y otras propiedades del proyecto como el nombre, descripción, y tipo de empaquetado (JAR o WAR).
- Una vez configuradas las opciones, Spring Initializr genera un proyecto con la estructura básica y un archivo de configuración y permite descargarlo como un archivo ZIP.

Índice

- Definición y origen
- Características principales
- Ecosistema
- MVC
- Maven
- **Configuración de un proyecto básico**
- Ejemplo

Configuración de un proyecto básico

- Directorio src/main/java: Contiene el código fuente de la aplicación, incluyendo la clase principal que inicia la aplicación Spring Boot.
- Directorio src/main/resources: Alberga recursos como archivos de propiedades (application.properties o application.yml) para la configuración de la aplicación, así como plantillas de vistas y archivos estáticos (CSS, JavaScript, imágenes).
- Directorio src/test/java: Destinado para el código de pruebas, donde se pueden escribir pruebas unitarias y de integración para la aplicación.

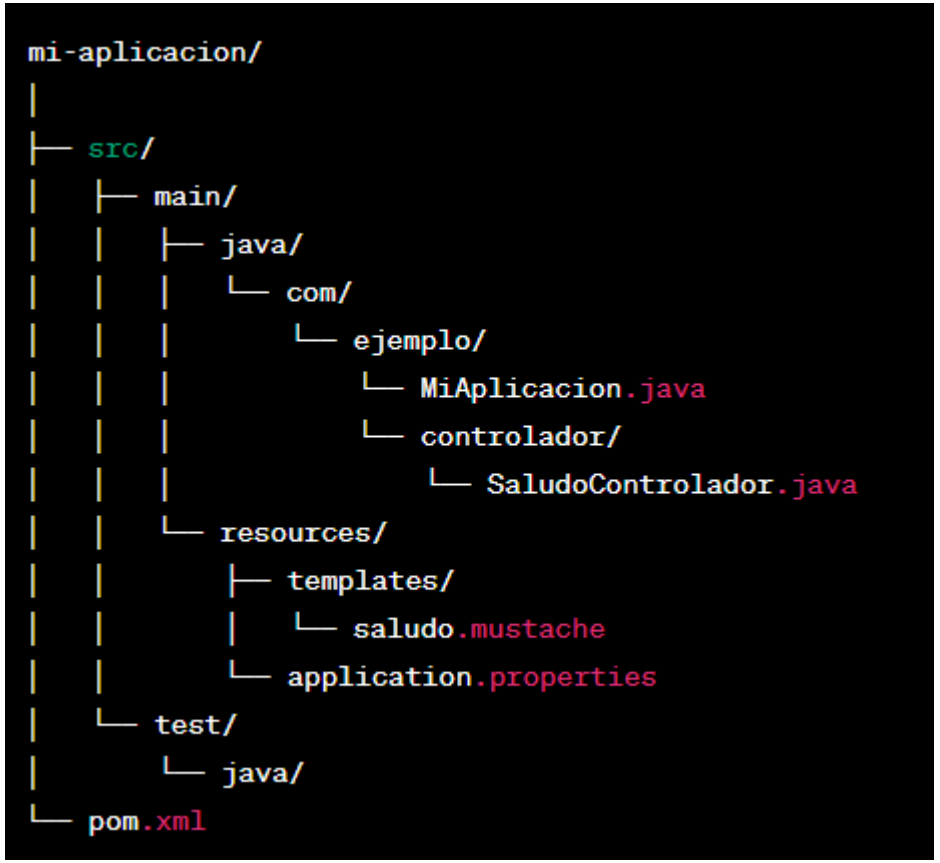
Configuración de un proyecto básico

- Archivo application.properties: Describe cómo estos archivos se utilizan para configurar diferentes aspectos de la aplicación, como configuraciones de base de datos, parámetros de servidor, y otras propiedades personalizadas.
- La clase principal de Spring Boot, se marca generalmente con las anotaciones @SpringBootApplication, que actúa como punto de entrada para la ejecución de la aplicación. Esta anotación incluye @Configuration, @EnableAutoConfiguration, y @ComponentScan.

Índice

- Definición y origen
- Características principales
- Ecosistema
- MVC
- Maven
- Configuración de un proyecto básico
- **Ejemplo**

Ejemplo



- Descripción de la Estructura:
  - src/main/java/: Este directorio contiene el código fuente de la aplicación.
  - com/ejemplo/MiAplicacion.java: La clase principal que arranca la aplicación Spring Boot.
  - com/ejemplo/controlador/SaludoControlador.java: Un controlador que maneja las solicitudes web y utiliza una vista Mustache para la respuesta.
  - src/main/resources/: Alberga los recursos de la aplicación.
  - templates/: Contiene las plantillas Mustache para las vistas.
  - saludo.mustache: La plantilla Mustache para la página de saludo.
  - application.properties: Archivo de configuración para propiedades de Spring Boot.
  - src/test/java/: Directorio para el código de prueba de la aplicación.
  - pom.xml: Archivo de configuración de Maven que incluye dependencias (como Spring Boot Starter Web y Spring Boot Starter Mustache) y configuración del proyecto.

Ejemplo

Project

- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language

- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot

- ○ 3.2.1 (SNAPSHOT)
- ● 3.2.0
- ○ 3.1.7 (SNAPSHOT)
- ○ 3.1.6

Project Metadata

Group	es.ssdd
Artifact	ejemplo1
Name	ejemplo1
Description	Hola mundo
Package name	es.ssdd.ejemplo1
Packaging	● Jar ○ War
Java	● 21 ○ 17

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

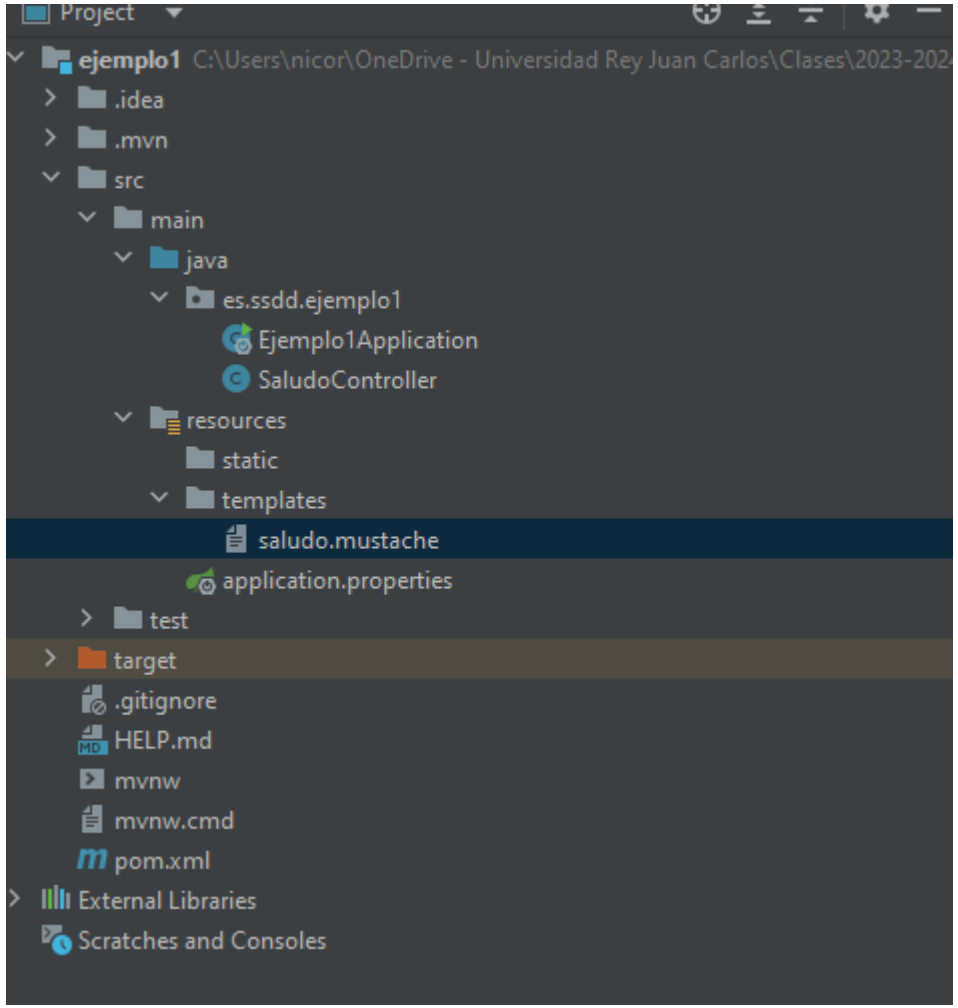
Mustache TEMPLATE ENGINES

Logic-less templates for both web and standalone environments. There are no if statements, else clauses, or for loops. Instead there are only tags.

https://start.spring.io/

Ejemplo

- Descomprimir el zip
- IntelliJ/Open/{ruta}
- Esperar la carga completa del proyecto



Ejemplo

SaludoControlador.java

```
package com.ejemplo.controlador;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
public class SaludoControlador {
        public String saludo(Model model) {
                model.addAttribute("mensaje", "Hola, Mundo!");
                return "saludo";
        }
}
```

- “mensaje” es una variable de Mustache
- A la izquierda, nombre de la variable
- A la derecha, contenido de la variable a mostrar por la plantilla
- “saludo” nombre de la vista

- ¡¡Correspondencias!!
  - Nombre de la vista – Nombre fichero
  - Nombre variable - Variable

http://localhost:8080/saludo

saludo.mustache

```
<!DOCTYPE html>
<html>
        <head> <title>Saludo</title>
        </head>
        <body>
                <h1>{{mensaje}}</h1>
        </body>
</html>
```

Sistemas Distribuidos

Bloque II

Desarrollo de sistemas distribuidos y aplicaciones web.

Tema 2.1:

Spring Framework

©2023 Autor Nicolás H. Rodríguez Uribe
Algunos derechos reservados
Este documento se distribuye bajo la licencia
"Atribución-CompartirIgual 4.0 Internacional" de Creative Commons, disponible en
https://creativecommons.org/licenses/by-sa/4.0/deed.es

Nicolás Rodríguez
nicolas.rodriguez@urjc.es