IDE´s para programar Spring Framework

IntelliJ IDEA

Visual Studio Code

Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.
Free. Open source.

Apache NetBeans IDE

IntelliJ IDEA

Spring Tool Suite

Eclipse

VSCode

Neatbeans

Configuración de Ambiente de Desarrollo

1. Instalación Jdk11 https://youtu.be/aoW2gsmYBHQ
2. Instalación IntelliJ IDEA https://www.jetbrains.com/idea/download/?section=
3. Crear cuenta https://github.com/ -- Instalación https://git-scm.com/downloads
4. Instalación de https://www.postman.com/
5. Mysql nstalación de https://dev.mysql.com/downloads/windows/installer/8.0.html
6. Posman

Repaso lenguaje Java

Principales Conceptos de Java:

Objeto

Clase

Interfaz

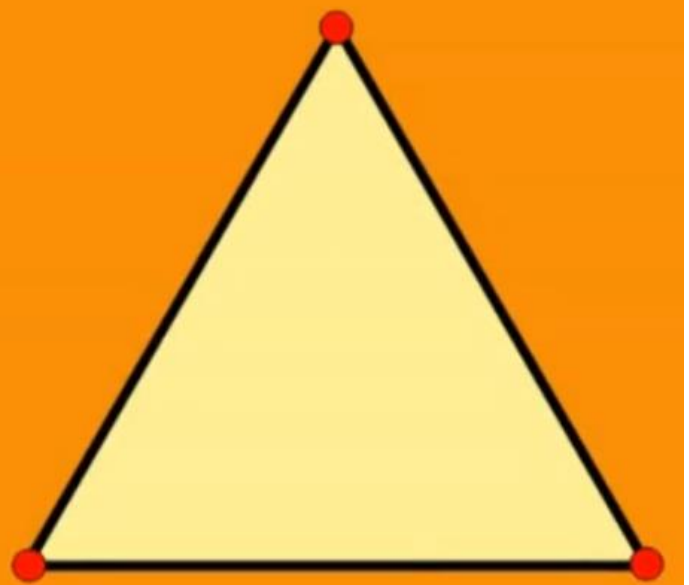
Herencia

Programación Orientada a Objetos

- Paradigma de programaión o forma de programar, donde se intenta representar objetos de la vida real.
- Todo se trata cómo un objeto, el cual tiene características y comportamientos.

Ejemplo de una clase:

```java
class Triangulo {
    private long base;
    private long altura;

    public Triangulo(long base, long altura) {
        this.base = base;
        this.altura = altura;
    }

    public long area() {
        return (base*altura)/2;
    }
}

Triangulo t1 = new Triangulo(2.0,3.0);
Triangulo t2 = new Triangulo(4.0,7.0);

t1.area(); // Área 3.0
t2.area(); // Área 14.0
```



Un objeto es:

- Es un elemento de software que intenta representar un objeto del mundo real.
- Un objeto tendrá sus propiedades y acciones a realizar.
- Estas propiedades y acciones están encapsuladas dentro del objeto (Principio de encapsulamiento).

Clase es:

- Instancia de un objeto
- Las clases representan los prototipos de los objetos que tenemos en el mundo real
- El nombre de una clase en Java siempre empieza con mayúscula.

CLASES Y OBJETOS

CLASE: es la abstracción de las propiedades y operaciones o acciones que ejecutan los objetos del mundo real.

OBJETO: son las instancias de una clase.

Los objetos que existen en el mundo real pueden ser conceptuales o físicos.

Cada objeto tiene valores propios para cada una de las características de la Clase.

Atributos:
- Color
- tamaño.
- Chasis

Comportamiento:
- mueve
- enciende motor.
-acelera.
- Frena

Ejemplo de variable:

```
int numero = 2;
String cadena = "Hola";
long decimal = 2.4;
boolean flag = true;
```

Las variables son utilizadas como propiedades dentro de los objetos.

```
class Triangulo {
    private long base;
    private long altura;
}
```

¿Qué son las variables en Java?

- Las variables Java son un espacio de memoria en el que guardamos un determinado valor (o dato)
- Para definir una variable seguiremos la estructura:

[privacidad] tipo_variable identificador;

Herencia es:

- La herencia es una forma de estructurar el software.
- Mediante la herencia podemos indicar que una clase hereda de otra. Es decir la clase extiende las capacidades (propiedades y métodos) que tenga y añade nuevas propiedades y acciones.

```
public class Triangulo extends Poligono {
    ...
}
```

```
public class Cuadrado extends Poligono {
    ...
}

public class Pentagono extends Poligono {
    ...
}
```

```
Triangulo t1 = new Triangulo(2.0,3.0);
t1.perimetro();
```

```
public class Poligono {

    private long[] lados;

    public Poligono(long[] lados) {
        this.lados = lados;
    }

    public long perimetro() {
        ...
    }

}
```

Interface es:

Un interface es una forma de establecer un contrato entre dos elementos

Un interface indica qué acciones son las que una determinada clase nos va a ofrecer cuando vayamos a utilizarla.

Cuando implementemos un interface (cuando lo usemos) deberemos de implementar todas las acciones (métodos) que este contenga.

```
public interface Vehiculo {
    public String matricula = "";
    public float maxVel
    public void arrancar();
    public void detener();
    default void claxon(){
      System.out.println("Sonando claxon");
    }
}
```

```
public class Coche implements Vehiculo {
    public void arrancar() {
        System.out.println("arrancando motor...");
    }
    public void detener() {
        System.out.println("deteniendo motor...");
    }
}
```

```
Vehiculo tesla = new Coche();

tesla.arrancar(); // arrancar el motor...
```

Spring

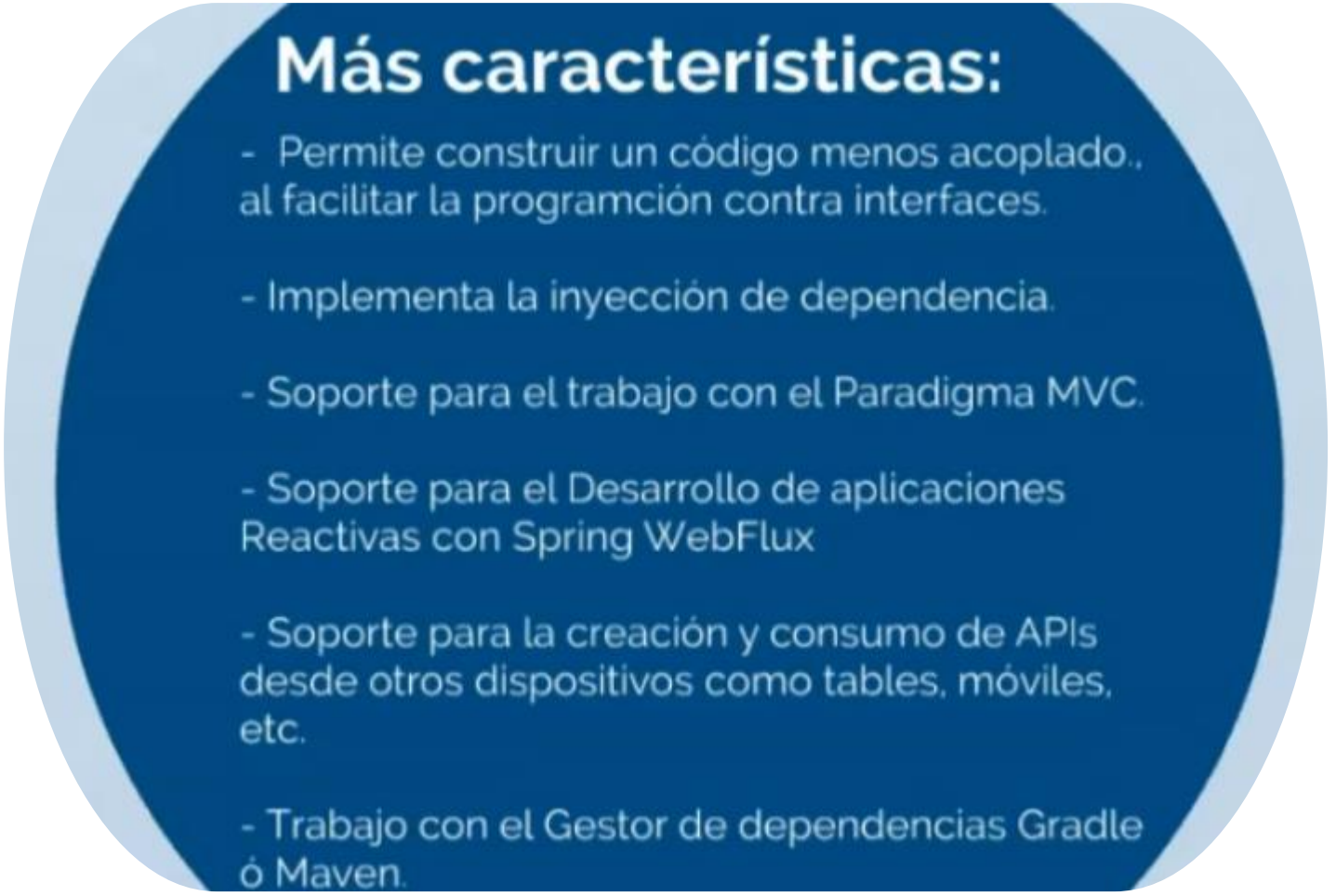
VS

Spring Boot

¿Qué es Spring?

- Spring es el Framework más popular para crear aplicaciones Java
  - *Pero...¿qué es un framework? Un framework es un "entorno de trabajo" compuesto por "reglas", y "herramientas" que facilitan enormemente el desarrollo de aplicaciones.*
- Spring es la alternativa al desarrollo de aplicaciones JEE.
  - *Más simple y más ligero*


spring
¿Qué es Spring Framework
- Es un framework opensource que facilita la creción de aplicaciones empresariales.
- Spring Framework está dividido en diversos módulos que podemos utilizar en distintas funcionalidades.
- Puedes usar cómo lenguaje de programación lava, Kotlin o Groovy.


Más características:
- Permite construir un código menos acoplado., al facilitar la programción contra interfaces.
- Implementa la inyección de dependencia.
- Soporte para el trabajo con el Paradigma MVC.
- Soporte para el Desarrollo de aplicaciones Reactivas con Spring WebFlux
- Soporte para la creación y consumo de APIs desde otros dispositivos como tables, móviles, etc.
- Trabajo con el Gestor de dependencias Gradle ó Maven.

Con Spring puedes:

- Sistemas web.

- Apis Rest.

- Trabajar con WebSockets.

- Integración con Base de Datos SQL y No SQL.

- Autenticación (Spring Security).

- Entre otras.

¿Qué es Spring Boot?

- Proyecto basado en Spring. No es lo mismo que Spring. Es un proyecto que forma parte del core de Spring, al igual que Spring Cloud, etc.
- El objetivo principal es que sólo te centres en correr la aplicación, sin preocuparte por temas de configuración, etc.
- Tiene la gran ventaja poder integrar librerías de terceros de manera muy sencilla.
- No tendremos que preocuparnos por configuraciones a nivel de XML, sólo configuraciones mínimas a nivel de properties (ponerle el puerto, etc).
- No tendremos que preocuparnos de desplegar nuestra aplicación en un servidor web local cuando queramos hacer pruebas, Spring Boot también contempla esto y lleva incorporado un servidor web dónde se desplegará nuestra aplicación automáticamente.

Características Spring Boot:

- Spring Boot permite compilar nuestras aplicaciones Web como un archivo .jar que podemos ejecutar como una aplicación Java normal.

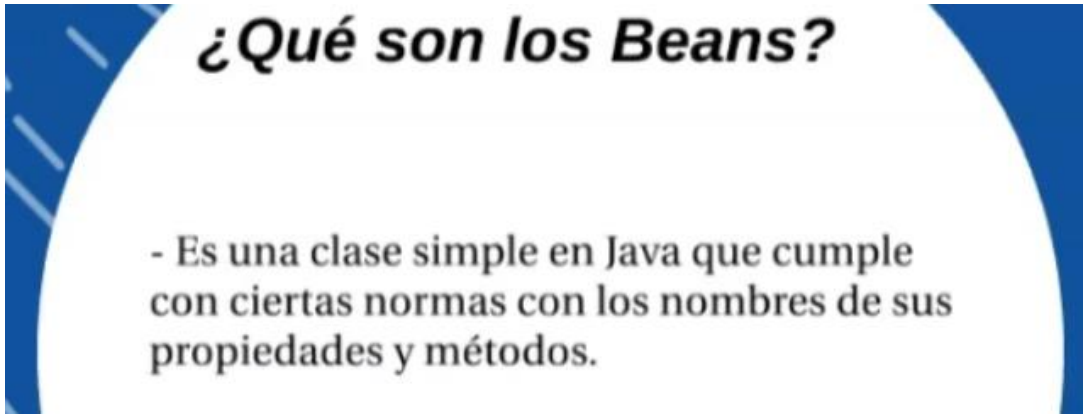
- Viene con un servidor web embebido (Tomcat por defecto).

- Spring Boot nos proporciona una serie de dependencias, llamadas starters, que podemos añadir a nuestro proyecto

- cualquier configuración puede ser modificada de ser necesario (puerto de arranque, por ejemplo).

Módulos de Spring Framework

CONCEPTOS DE SPRING



Condiciones para un Bean:

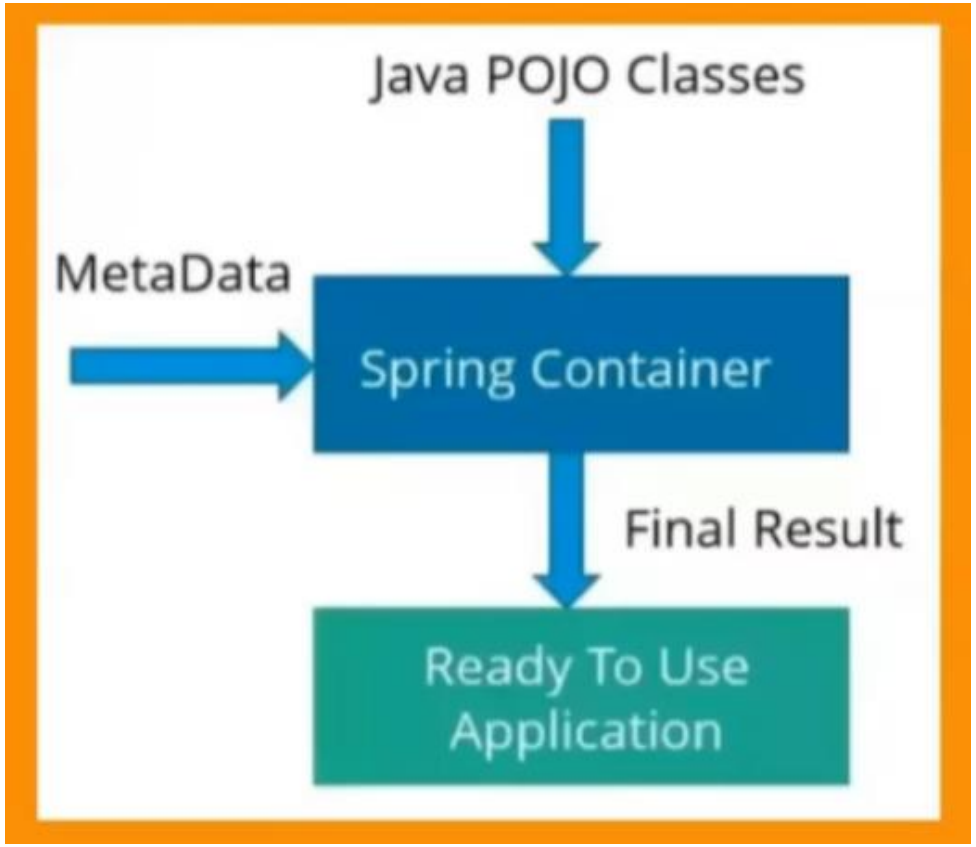
- **Constructor sin argumentos**: aunque ya existe por defecto (no se ve pero está), se aconseja el declararlo.
- **Atributos privados.**
- **Getters y Setters de todos los atributos.**
- **El bean puede implementar Serializable.**
- **Requieren una anotación o crear un fichero .xml donde se detalle que es un Bean.**

CONCEPTOS DE SPRING

Spring Container es:

- Dentro de la infraestructura de Spring tenemos Spring Container (contendedor), que vendría siendo el núcleo del framework.

- La aplicación de Spring, puede leer Beans mediante a distintas fuentes. Esta lectura se guardará en lo que conocemos como Spring Container.



El contenedor creará los objetos, los conectará, los configurará y administrará su ciclo de vida completo desde la creación hasta la destrucción

CONCEPTOS DE SPRING

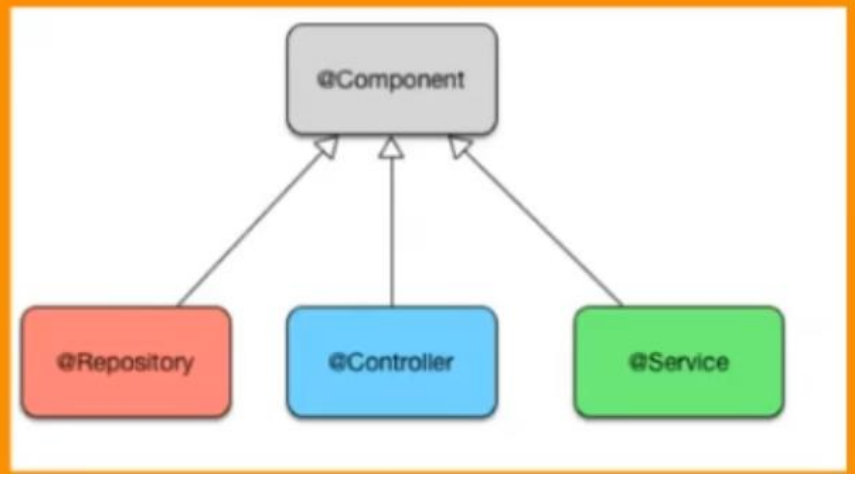
Ejemplo de Bean:

```java
import java.io.Serializable;
import org.springframework.stereotype.Component;

@Component
public class Persona implements Serializable{

    private static final long serialVersionUID = 6821412775269078235L;

    private String nombre;
    private String direccion;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getDireccion() {
        return direccion;
    }
    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

}
```

CONCEPTOS DE SPRING

Spring Stereotypes y anotaciones:

Spring define un conjunto de anotaciones core que categorizan cada uno de los componentes asociandoles una responsabilidad concreta.



CONCEPTOS DE SPRING

¿Qué hace la anotación @component?
@Component es una anotación que permite a Spring detectar automáticamente nuestros beans personalizados .

En otras palabras, sin tener que escribir ningún código explícito, Spring: Analizará nuestra aplicación en busca de clases anotadas con @Component. Cree una instancia de ellos e inyecte cualquier dependencia especificada en ellos

Tipos de Stereotypes:

@Component: Es el estereotipo general y permite anotar un bean para que Spring lo considere uno de sus objetos.

@Repository: Es el estereotipo que se encarga de dar de alta un bean para que implemente el patrón repositorio que es el encargado de almacenar datos en una base de datos o repositorio de información que se necesite. Al marcar el bean con esta anotación Spring aporta servicios transversales como conversión de tipos de excepciones.

@Repository: especificamos una clase con @Repository para indicar que se trata de operaciones CRUD , por lo general, se utiliza con DAO (objeto de acceso a datos) o implementaciones de repositorio que se ocupan de tablas de base de datos.

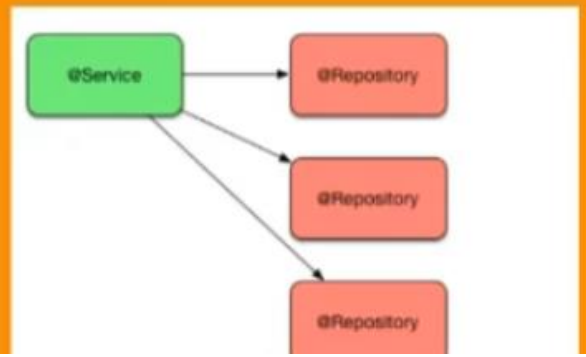
CONCEPTOS DE SPRING

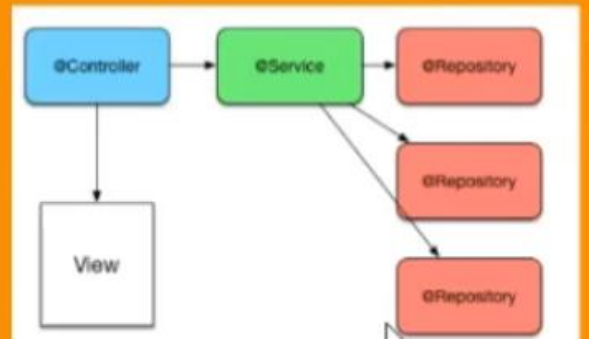
@Service: especificamos una clase con @Service para indicar que mantienen la lógica empresarial. Además de usarse en la capa de servicio, no hay ningún otro uso especial para esta anotación. Las clases de utilidad se pueden marcar como clases de servicio.

Tipos de Stereotypes:

@Service : Este estereotipo se encarga de gestionar las operaciones de negocio más importantes a nivel de la aplicación y puede llamar a varios repositorios de forma simultánea. Su tarea fundamental es la de agregador.

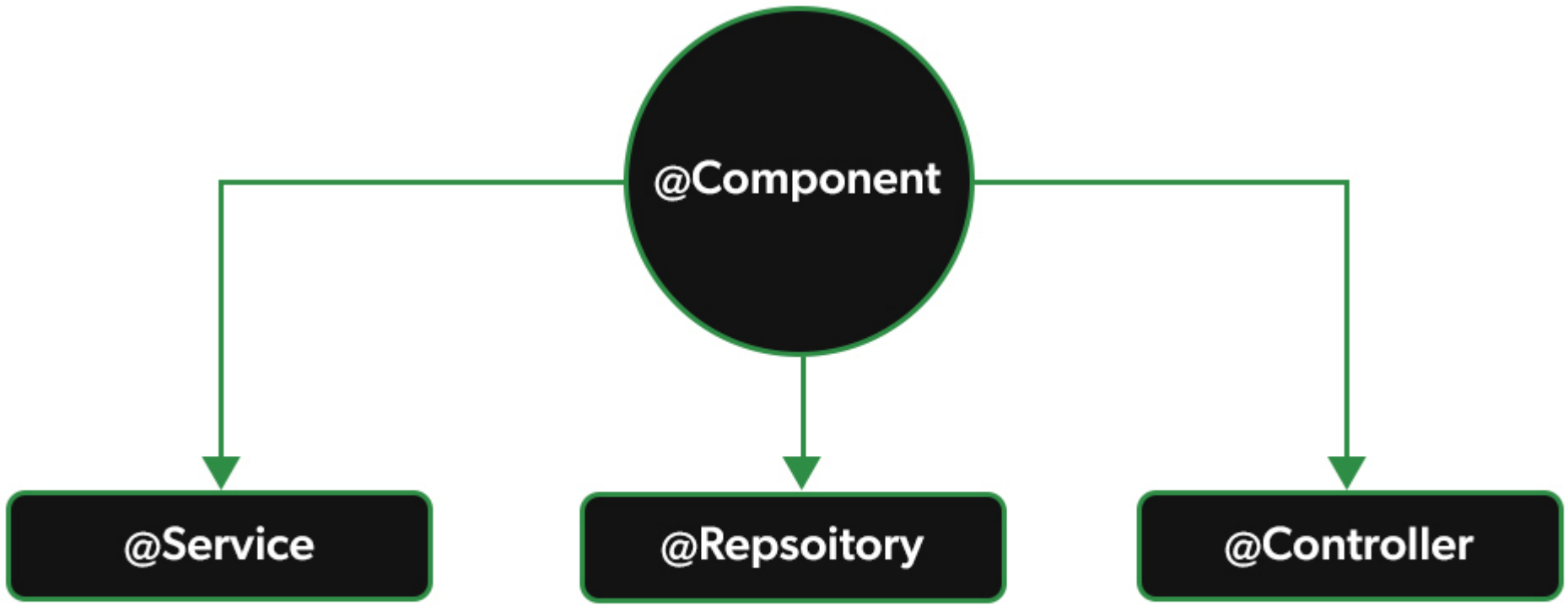
@Controller: El último de los estereotipos que es el que realiza las tareas de controlador y gestión de la comunicación entre el usuario y el aplicativo.





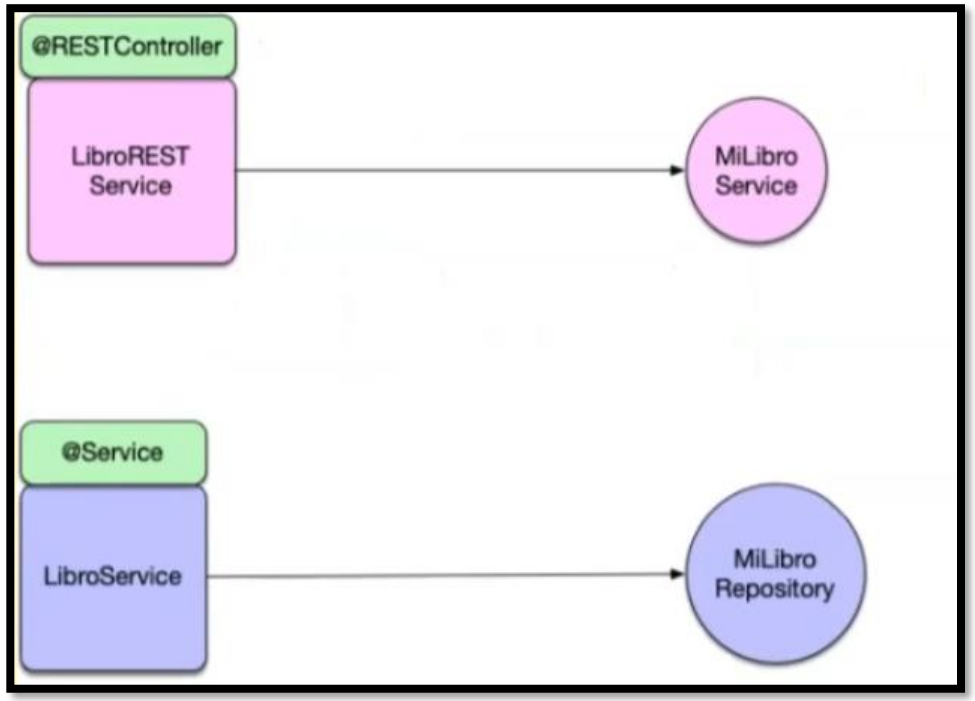
@Controller: especificamos una clase con @Controller para indicar que son controladores frontales y responsables de manejar las solicitudes de los usuarios y devolver la respuesta adecuada. Se utiliza principalmente con servicios web REST.

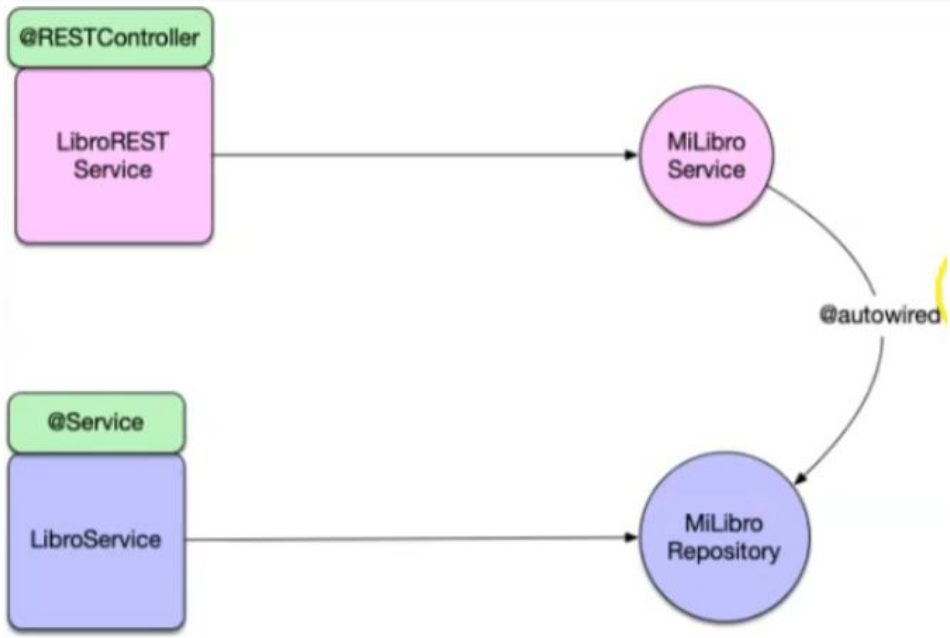
Entonces, las anotaciones de estereotipos en Spring son ***@Component, @Service, @Repository y @Controller*** .



Inyección de Dependencia:

Spring **@Autowired** es una de las anotaciones más habituales cuando trabajamos con Spring Framework ya que se trata de la anotación que permite inyectar unas dependencias con otras dentro de Spring .





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

import org.springframework.stereotype.Repository;

import com.company.books.backend.model.Libro;

@Repository
public class LibroRepository {
  public List<Libro> buscarTodos() {
    List<Libro> lista= new ArrayList<Libro>();
    lista.add(new Libro ());
    lista.add(new Libro ());

    return lista;

  }

}
```

```java
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class LibroRestService {

  @Autowired
  LibroRepository repositorio;

  @RequestMapping("/libros")
  public List<Libro> buscarTodos() {

    return repositorio.buscarTodos();

  }
}
```

¿Que es una API?

- Interfaz de programación de aplicaciones.

- Es un conjunto de requisiciones que permite la comunicación de datos entre aplicaciones.

- la API utiliza requisiciones HTTP responsables de las operaciones básicas necesarias para la manipulación de datos.

Las principales solicitudes

- POST: crea datos en el servidor;

- GET: lectura de datos en el host;

- DELETE: borra la información;

- PUT: registro de actualizaciones.

Rest es:

- La abreviación de Representational State Transfer

- Es un conjunto de restricciones que se utilizan para que las solicitudes HTTP cumplan con las directrizes definidas en la arquitectura.

Restricciones determinadas por Rest son:

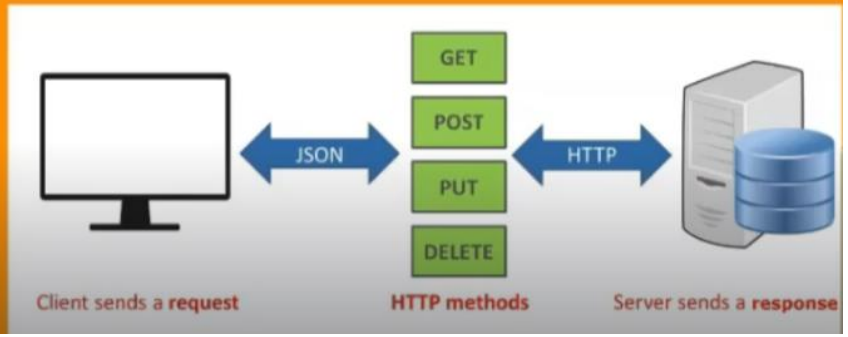
-**Cliente-servidor**: las aplicaciones existentes en el servidor y el cliente deben estar separadas.
- **Sin estado**: las requisiciones se realizan de forma independiente, es decir, cada una ejecuta solo una determinada acción.
- **Caché**: la API debe utilizar la caché para evitar llamadas recurrentes al servidor.

Entonces API Rest:

significa utilizar una API para acceder a aplicaciones back-end, de manera que esa comunicación se realice con los estándares definidos por el estilo de arquitectura Rest.

Las APIs sirven:

para comunicarse entre aplicaciones para intercambiar informaciones d forma rápida y segura.

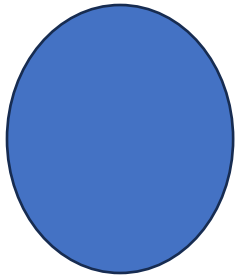


Ventajas uso de APIS:

- Separación entre cliente y servidor.

- Multiplataforma (Las requisiciones HTTP realizadas en API Rest devuelven datos en formato JSON).

- Sus características permiten la integración con diferentes aplicaciones; entre ellos, redes sociales y sistemas de pago



jsonplaceholder.typicode.com/users


```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
```


¿Que es JSON?

- JSON significa JavaScript Object Notation o Notación de Objetos de JavaScript.

- Formato para el intercambio de datos entre aplicaciones.

- Es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

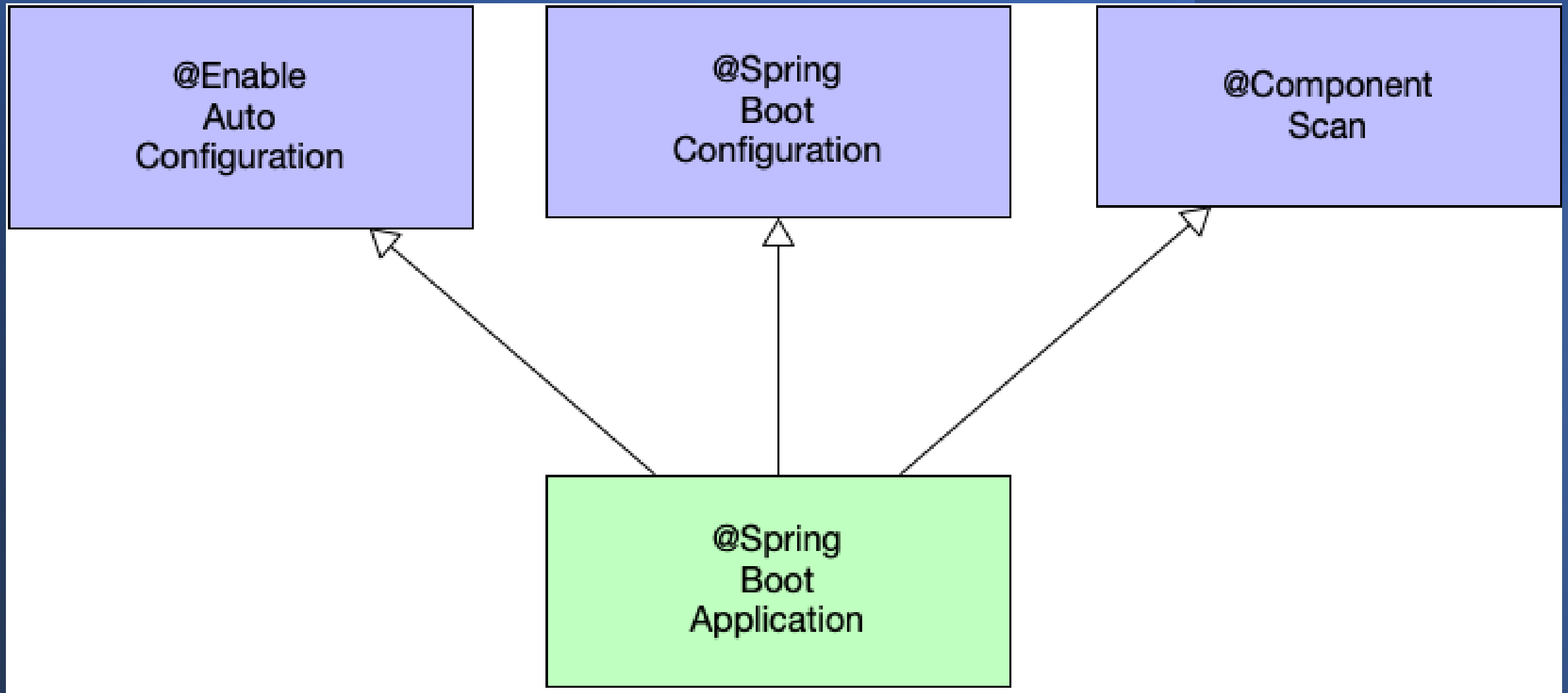
- JSON es un formato de texto completamente independiente de lenguaje

Caracteristicas de JSON:

- JSON es solo un formato de datos.

- Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.

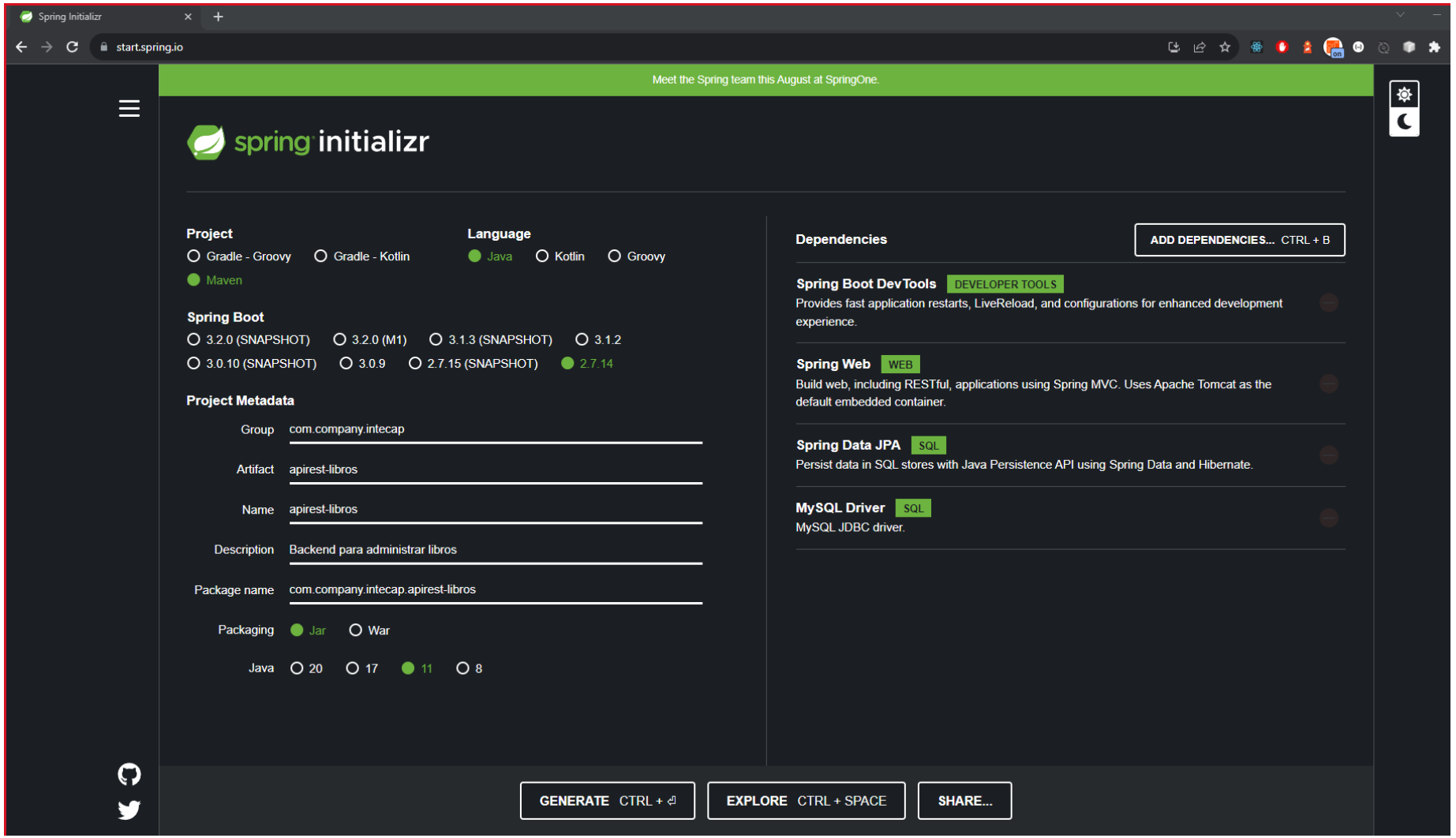
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.



Herencia entre anotaciones

Primer Proyecto de Spring Boot

1. Inicializador del Proyecto. https://start.spring.io/



https://start.spring.io/#!type=maven-project&language=java&platformVersion=2.7.14&packaging=jar&jvmVersion=11&groupId=com.company.intecap&artifactId=apirest-libros&name=apirest-libros&description=Backend%20para%20administrar%20libros&packageName=com.company.intecap.apirest-libros&dependencies=devtools,web,data-jpa,mysql

Bibliografía

https://www.arquitecturajava.com/springbootapplication-una-anotacion-clave/

https://hackernoon.com/es/la-diferencia-entre-jdbc-jpa-hibernate-y-spring-data-jpa

https://es.slideshare.net/angenio2/programacin-3-clases-y-objetos-en-java

Preguntas de entrevista Spring Boot
https://java2blog.com/spring-interview-questions-and-answers/?ref=hackernoon.com

Curso de Java
https://www.w3schools.com/java/
https://youtu.be/2ZXiuh0rg3M?list=PLWtYZ2ejMVJkjOuTCzIk61j7XKfpIR74K

https://www.freecodecamp.org/espanol/news/interfaces-java-explicadas-con-ejemplos/

https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/142#:~:text=Spring%20se%20basa%20en%20IoC,la%20creaci%C3%B3n%20de%20estos%20objetos.

https://www.tutorialspoint.com/spring/spring_ioc_containers.htm#:~:text=The%20Spring%20container%20is%20at,that%20make%20up%20an%20application.

https://www.geeksforgeeks.org/spring-stereotype-annotations/